UNIVERSIDAD POLITÉCNICA DE MADRID
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA
Y SISTEMAS DE TELECOMUNICACIÓN
Campus Sur. Ctra. de Valencia km. 7. 28031 Madrid

PROYECTO FIN DE CARRERA
PLAN 2000

TEMA: Ontologías para el desarrollo de servicios SOA.

TÍTULO: Integración de SSN con una ontología de servicios.

AUTOR: Susana Ortiz Cabañas

TUTOR: Rubén de Diego Martínez **Vº Bº**.

DEPARTAMENTO: Diatel

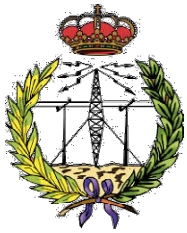
Miembros del tribunal calificador

PRESIDENTE: Manuel Labrador Pretel
VOCAL: Rubén de Diego Martínez
VOCAL SECRETARIO: Gregorio Rubio Cifuentes
DIRECTOR:

Fecha de lectura:

Calificación: **El Secretario,**

El principal objetivo de este Proyecto de Fin de Carrera es la generación semiautomática de código a partir de un modelo de sistemas descrito en función de las clases y propiedades proporcionadas por la ontología SSN y representado en un fichero de tipo OWL.
SSN fue especialmente diseñada para diseñada para la descripción semántica de sensores y las redes de las que forman parte con el fin de permitir una mejor interacción entre los dispositivos y los sistemas que hacen uso de ellos.
El trabajo desarrollado a lo largo de este proyecto se ha dividido en varias partes. Primero se realiza un análisis de la ontología mencionada. A continuación se describe un sistema simulado de sensores y por último se implementan las aplicaciones para la generación automática de interfaces y la representación gráfica de los dispositivos del sistema.



UNIVERSIDAD POLITÉCNICA DE MADRID

Escuela Técnica Superior de Ingeniería y

Sistemas de Telecomunicación

PROYECTO FIN DE CARRERA

Integración de SSN con una Ontología de Servicios

AUTOR

Susana Ortiz Cabañas

TUTOR

Rubén De Diego Martínez

Septiembre de 2014

RESUMEN

Los servicios en red que conocemos actualmente están basados en documentos y enlaces de hipertexto que los relacionan entre sí sin aportar verdadera información acerca de los contenidos que representan. Podría decirse que se trata de "una red diseñada por personas para ser interpretada por personas".
El objetivo principal de los últimos años es encaminar esta red hacia una web de conocimiento, en la que la información pueda ser interpretada por agentes computerizados de manera automática.
Para llevar a cabo esta transformación es necesaria la utilización de nuevas tecnologías especialmente diseñadas para la descripción de contenidos como son las ontologías.

Si bien las redes convencionales están evolucionando, no son las únicas que lo están haciendo. El rápido crecimiento de las redes de sensores y el importante aumento en el número de dispositivos conectados a internet, hace necesaria la incorporación de tecnologías de la web semántica a este tipo de redes.

Para la realización de este Proyecto de Fin de Carrera se utilizará la ontología SSN, diseñada para la descripción semántica de sensores y las redes de las que forman parte con el fin de permitir una mejor interacción entre los dispositivos y los sistemas que hacen uso de ellos.

El trabajo desarrollado a lo largo de este Proyecto de Fin de Carrera gira en torno a esta ontología, siendo el principal objetivo la generación semiautomática de código a partir de un modelo de sistemas descrito en función de las clases y propiedades proporcionadas por SSN.
Para alcanzar este fin se dividirá el proyecto en varias partes. Primero se realizará un análisis de la ontología mencionada. A continuación se describirá un sistema simulado de sensores y por último se implementarán las aplicaciones para la generación automática de interfaces y la representación gráfica de los dispositivos del sistema a partir de la representación del éste en un fichero de tipo OWL.

ABSTRACT

The web we know today is based on documents and hypertext links that relate these documents with each another, without providing consistent information about the contents they represent. It could be said that it's a network designed by people to be used by people.
The main goal of the last couple of years is to guide this network into a web of knowledge, where information can be automatically processed by machines.
This transformation, requires the use of new technologies specially designed for content description such as ontologies.

Nowadays, conventional networks are not the only type of networks evolving. The use of sensor networks and the number of sensor devices connected to the Internet is rapidly increasing, making the use the integration of semantic web technologies to this kind of networks completely necessary.

The SSN ontology will be used for the development of this Final Degree Dissertation. This ontology was design to semantically describe sensors and the networks they're part of, allowing a better interaction between devices and the systems that use them.

The development carried through this Final Degree Dissertation revolves around this ontology and aims to achieve semiautomatic code generation starting from a system model described based on classes and properties provided by SSN.

To reach this goal, de Dissertation will be divided in several parts. First, an analysis about the mentioned ontology will be made. Following this, a simulated sensor system will be described, and finally, the implementation of the applications will take place. One of these applications will automatically generate de interfaces and the other one will graphically represents the devices in the sensor system, making use of the system representation in an OWL file.

Tabla de contenido

Tabla de figuras

Acrónimos

CSIRO: Commonwealth Scientific and Industrial Research Organisation, 16

DnS: Descriptions and Situations, 19

DOLCE: Descriptive Ontology for Linguistic and Cognitive Engineering, 19

DUL: DOLCE Ultra Lite, 19

HTML: HyperText Markup Language, 2

MMI: Marine Metadata Interoperability, 16

OWL: Web Ontology Language, 3

PFC: Proyecto de Fin de Carrera, 3

RDF: Resource Description Framework, 3

SDO: Sensor Data Ontology, 16

SemSorGrid4Env: Semantic Sensor Grids for Environmental Applications, 30

SPITFIRE: Semantic-Service Provisioning for the Internet of Things, 29

SSN: Semanic Sensor Network, 1

SSN-XG: Semantic Sensor Networks Incubator Group, 15

SSO: Stimulus-Sensor-Observation, 18

SWAMO: Sensor Web for Autonomous Mission Operations, 16

URI: Uniform Resource Identifier, 11

W3C: World Wide Web Consortium, 15

XML: Extensible Markup Language, 3

1. Introducción y objetivos

1.1 Contextualización.

El creciente interés de los últimos años en el uso de redes de sensores viene acompañado de un importante aumento en la cantidad de datos recogidos y de un amplio abanico de dispositivos y procedimientos de medición [1]. Esto, junto con la ausencia de estandarización y comunicación entre sistemas y redes plantea un problema: se tienen demasiados datos y muy poca información.

Con el objetivo de poner fin a esta tesitura, se están incorporando diversas tecnologías a las redes de sensores que permiten una mejor interacción entre los dispositivos y los sistemas que hacen uso de ellos.

Puesto que el elemento más importante de este proyecto de fin de carrera es la ontología *Semantic Sensor Network* (SSN) [2], cuyo objetivo principal es dotar de semántica a las redes de sensores y conseguir transformar grandes cantidades de

datos en conocimiento, se ha considerado conveniente hacer mención a la red semántica, antes de detenerse a definir el concepto general de ontología, que se verá más adelante.

Los servicios en red que conocemos actualmente fueron diseñados con la intención de tener una fácil comprensión y lectura para las personas, no con el objetivo de que la información de la que hacen uso pudiera ser procesada de manera automática [3].

Sin embargo, la tendencia actual es diseñar modelos y estandarizar lenguajes y herramientas con el fin de transformar esta red y convertirla en un espacio de conocimiento, dando lugar a lo que se ha denominado *web semántica*.

Esta red semántica no pretende definir una nueva web, sino que busca ser una extensión de la web actual, dotada de un mayor significado.

Su objetivo principal es conseguir que los contenidos de la red puedan ser interpretados tanto por agentes humanos como por agentes computerizados, lo que permitiría a los usuarios buscar y compartir información de manera más rápida y sencilla.

Tim Berners-Lee, creador de la World Wide Web, e impulsor de la web semántica, define ésta como: [4]

"Una web de datos que puedan ser procesados directa e indirectamente por máquinas."

Para lograr este fin, se hace imprescindible integrar en la web un conjunto de metadatos que sean comprensibles para los ordenadores y que aporten información detallada de los contenidos, su significado y la forma en la se interrelacionan las páginas y los datos entre sí.

Hoy en día, un gran número de páginas que forman la web están escritas en HTML *(HyperText Markup Language)*, un lenguaje que no permite definir los datos que representa ni las relaciones que tienen unos con otros.
El hecho de que no se tenga información acerca de los contenidos es lo que provoca que en ocasiones, al realizar búsquedas por internet acerca de un tema concreto, los resultados obtenidos no sean los esperados. Considérese un ejemplo en el que se necesita información sobre el color naranja. Al introducir el término "naranja" en un motor de búsqueda podría ocurrir que se obtuvieran resultados acerca del color del mismo nombre. Añadiendo información semántica a los recursos esto no ocurriría, pues estaría bien definido a qué ámbito concreto están vinculados los datos que se representan en una web.

Para solventar las carencias que padece HTML, la web semántica hace uso de lenguajes especialmente diseñados para la descripción de contenidos: RDF *(Resource Description Framework)*, OWL *(Web Ontology Language)* y XML *(Extensible Markup Language)*.

Combinando estas tecnologías, que serán descritas más adelante, es posible dotar con una lógica y un significado a cada recurso de la red, permitiendo a los ordenadores conocer la información que manejan con el fin de que pueda ser utilizada de manera eficiente y semiautomática.

A raíz de esta web y con la incorporación a internet de millones de dispositivos cargados de sensores como puede ser un Smartphone, que recogen información del entorno y de los propios usuarios, surge la red semántica de sensores.

1.2 Objetivos.

El propósito principal de este Proyecto Fin de Carrera (PFC) es la implementación de una aplicación que utilizando la ontología *Semantic Sensor Network* (SSN) sea capaz de representar un sistema de sensores simulado y generar las interfaces java de los elementos que lo componen de forma automática.

Para llevar a cabo este trabajo se han definido los siguientes objetivos:

- Realizar un estudio de la ontología SSN su aplicación en un escenario real con el fin de elaborar una descripción semántica del sistema y sus dispositivos lo más precisa posible.

- Familiarizarse con las diversas herramientas que se utilizarán para llevar a cabo la implementación de los programas que conforman el proyecto, en especial la librería Jena, un entorno de desarrollo Java diseñado para el creación de aplicaciones semánticas.

- Una vez se hayan adquirido los conocimientos necesarios, se realizará la definición cada uno de los componentes del sistema simulado a partir de las clases y propiedades propuestas por SSN.

- Implementar una aplicación capaz de generar las interfaces de los dispositivos de un sistema de sensores de manera semiautomática a partir de un fichero OWL en el que se hayan modelado sus componentes de acuerdo a la ontología SSN.

- Desarrollar un servidor a partir de las interfaces generadas que permita representar gráficamente los elementos que componen el sistema de sensores.

1.3 Estructura de la memoria.

A continuación se presenta una breve descripción de los capítulos en los que se divide el resto de la memoria de este PFC:

- **Capítulo 2:** En este capítulo, se hará una presentación del Estado del Arte, en el que se describirá del concepto de ontología, enumerando sus principales elementos, funciones y características. También se hará una pequeña introducción acerca del lenguaje OWL en el que está escrita la ontología SSN, y se hará un estudio detallado de ésta, con el fin de utilizarla de la forma más adecuada posible en la descripción del sistema de sensores simulado con el que se va a trabajar.

- **Capítulo 3:** El capítulo 3 corresponde a la especificación de dicho sistema de sensores. A partir de las posibilidades que ofrece SSN para la descripción de dispositivos y los despliegues en los que se encuentran, se elegirán algunas de estas propiedades para caracterizar el entorno de trabajo con un nivel de detalle no demasiado profundo, pero suficiente para el desarrollo de este PFC, en el que la compresión de la ontología y su integración en una aplicación es el principal objetivo.

- **Capítulo 4:** En esta sección se va a definir el diseño la aplicación que se ha implementado y que hace uso del modelo definido en el capítulo anterior. Además de la aplicación para la representación del sistema de sensores se define también un pequeño programa que se ha propuesto, con el fin de facilitar la escritura de los dispositivos de acuerdo a la ontología SSN.

- **Capítulo 5:** Este último capítulo recoge las conclusiones que pueden obtenerse de la memoria y las posibles líneas de actuación para un trabajo futuro.

2 Estado del arte

A lo largo de este capítulo se van a definir algunos conceptos importantes sobre los que se ha hecho mención y que requieren una explicación más detallada.

2.1 Ontologías.

Las ontologías son una pieza clave dentro de la web semántica para la representación de la información y el conocimiento. Pero ¿qué son exactamente?

Existen multitud de definiciones de este término, derivado de la filosofía, sin embargo en el ámbito en el que se desarrolla este PFC se puede definir la ontología como un sistema que permite representar el conocimiento de tal manera que sea legible para los ordenadores, esté consensuado y sea además reutilizable.

Las ontologías proporcionan un vocabulario para definir los conceptos y entidades de algún dominio o área de conocimiento, cómo deben ser agrupadas y qué relaciones las conectan entre sí.

A continuación se describen brevemente algunos de los elementos más importantes que las componen:

- **Clases o conceptos**: son las ideas básicas que se quieren formalizar. Pueden ser tipos de objetos, métodos, procesos de razonamiento, etc.

- **Instancias o *individuals***: se utilizan para representar objetos determinados de un concepto o clase.

- **Restricciones:** son descripciones formales que deben cumplir ciertos datos para poder ser aceptados como válidos.

- **Axiomas**: son teoremas que se declaran sobre relaciones que deben cumplir los elementos de la ontología. Por ejemplo: "*Si A y B son de la clase C, entonces A no es subclase de B*", "*Para todo A que cumpla la condición C1, A es B*", etc.
  Los axiomas, junto con la herencia de conceptos, permiten inferir conocimiento que no esté indicado explícitamente.

- **Relaciones o propiedades**: representan la interacción y enlace entre los conceptos e instancias del dominio. Por ejemplo: *subclase-de*, *parte-de*, *conectado-a*, etc.

Según [5], existen dos conceptos importantes que pueden indicarse a la hora de crear una propiedad: el rango de ésta y su dominio.

El dominio lo componen los recursos que contarán con la propiedad creada, y el rango hace referencia a los valores que esa propiedad puede tomar.

Si una propiedad no indica su dominio, no se puede inferir información acerca de los recursos que describe, ya que todas las clases e instancias de la ontología podrán utilizarla. Por el contrario, si el dominio queda definido por un tipo se puede

asegurar, que todos los elementos que contengan esa propiedad serán del tipo del dominio.
Lo mismo ocurre con el rango y el valor de la propiedad.

En el ejemplo de la Figura 2-1 se puede observar como una propiedad, "tieneColorDeOjos", con un dominio definido por las clases "Animal" y "Persona" y un rango especificado por la clase "Color".

Si se creara una instancia "Jaime" y se estableciera que cuenta con la propiedad "tieneColorDeOjos" y que además el valor de dicha propiedad es "Azul", se podría asegurar, fijándose únicamente en el dominio y el rango de la propiedad, que "Jaime" es "Animal" y "Persona" y que "Azul" es un "Color".

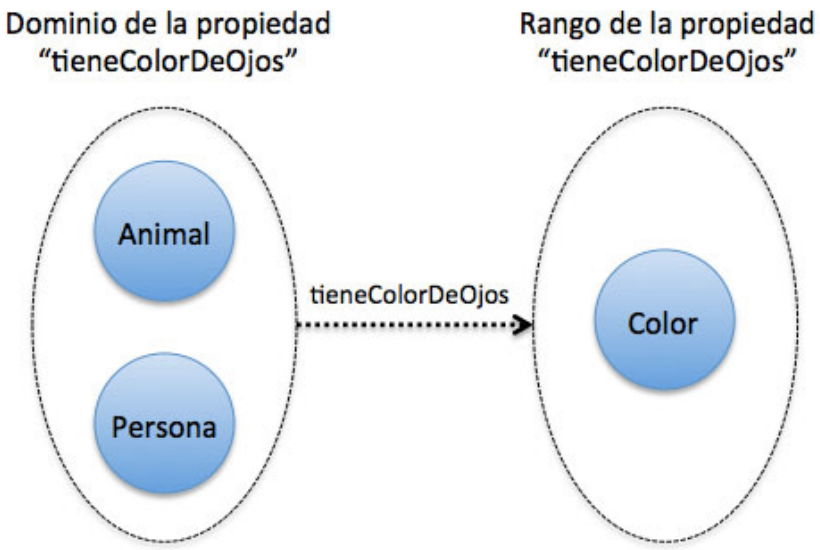


Figura 2-1: Rango y dominio de una propiedad

Es de gran importancia a la hora de diseñar una ontología, la cantidad de información a incluir en el modelo. Cuanto más precisa sea ésta, mayor será el potencial de entendimiento acerca de los datos que representa y de cómo éstos pueden ser usados. Por el contrario, si se elabora algo excesivamente complejo, con categorías y

relaciones que no sean estrictamente necesarias, puede resultar demasiado complicada de utilizar.

Quizás, la tarea más difícil aparece cuando se ha de establecer si debe modelarse un objeto como *individual* de una clase general o por el contrario es más conveniente definir una subclase específica y crear una instancia a partir de ella. Si bien ambas disposiciones son correctas, el significado que se le da al objeto dentro de la ontología es diferente en función de la opción por la que se opte.

Para comprender esto de manera sencilla, se propone analizar el siguiente ejemplo en el que se tiene una clase llamada "Mamíferos" y quiere modelarse un León. ¿Es "León" una instancia de la clase "Mamíferos" o debe definirse una subclase de esta categoría? Para establecer que afirmación es más correcta en cada situación, hay que tener en cuenta el contexto global de la ontología. En el supuesto caso en el que se estuviera modelando el reino animal en su conjunto, bastaría con definir un león como un *individual* de la clase "Mamífero", tomándolo como un tipo de animal mamífero concreto. Sin embargo, va a suponerse ahora que la ontología quiere utilizarse para representar los animales que hay en un zoológico. En este caso podría contarse con distintos individuos de esta especie, con características como peso, edad y tamaño diferentes y relevantes, por lo que lo más acertado para este supuesto sería crear una subclase "León" a partir de la clase "Mamífero", de la cual se crearían las diferentes instancias.

No se debe olvidar, que uno de los propósitos de las ontologías es que sean reutilizables, por lo que se pueden añadir clases y propiedades a un modelo sencillo ya existente y adaptarlo hacia otro más complejo según sea necesario.

2.2 OWL: Web Ontology Language

Existen varios lenguajes para la descripción de ontologías, pero uno de los más importantes, y el más extendido en la actualidad es el *Web Ontology Language*, en su acrónimo, OWL [6].

Antes de entrar a ver más detalles sobre este lenguaje, es interesante explicar brevemente algunos conceptos asociados a la web semántica y las ontologías a los que se ha hecho referencia y que se encuentran estrechamente ligados a este lenguaje de marcado.

En primer lugar es importante definir el concepto de "tripletas", unas expresiones de tipo sujeto-predicado-objeto en las que RDF se basa para hacer las declaraciones sobre los diferentes recursos. El sujeto se corresponde con el recurso que se está describiendo, el predicado constituye la propiedad o relación que se establece acerca del sujeto, y por último el objeto, que es o bien el valor de dicha propiedad, o un recurso diferente con el que se establece la relación.

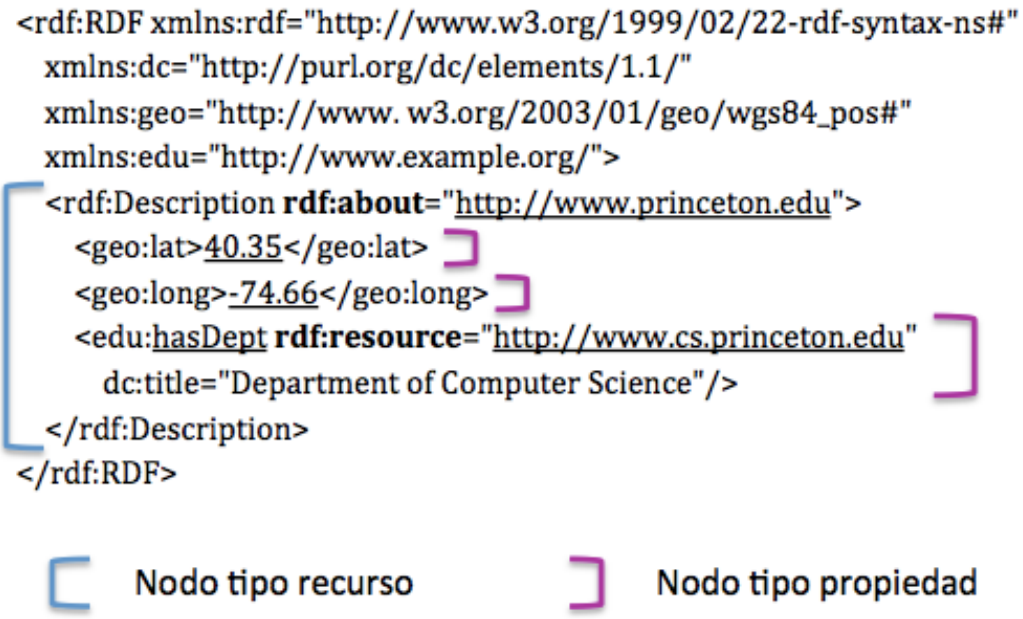
Los nombres de los recursos son URI (*Uniform Resource Identifier*) que generalmente tienen el formato de una dirección web a pesar de no serlo. Únicamente se utilizan como identificadores de los recursos dentro del documento RDF.

Estos documentos pueden exportarse en diferentes formatos, siendo RDF/XML el más común. XML es un lenguaje de marcas que define una serie de normas para estructurar documentos de tal forma que estos sean legibles tanto para personas como para máquinas. Permite etiquetar la información que se maneja, pero sin aportar significado.

Según [7], en un documento XML/RDF existen dos tipos de nodos, de recurso y de propiedad. Los nodos de tipo recurso corresponden a los sujetos y los objetos de las tripletas y generalmente cuentan con el atributo *rdf:about* que contiene la URI del

recurso que representan. Los nodos de tipo propiedad contienen valores literales o una referencia a un recurso (objeto de la tripleta) mediante el atributo *rdf:resource*.

En la Figura 2-2 se muestra un extracto de un fichero RDF/XML en el que pueden diferenciarse los dos tipos de nodos y atributos mencionados.



Fuente: http://www.xml.com/pub/a/2001/01/24/rdf.html?page=2

Figura 2-2: Nodos de un documento RDF/XML

RDF puede ser ampliado a través de un vocabulario, RDF Schema, que proporciona numerosas formas de describir las propiedades y tipos de recursos de RDF, e incluso permite introducir jerarquías mediante la definición de clases y subclases. Sin embargo, para cumplir los requerimientos de la web semántica y que la información contenida en documentos pueda ser procesada e interpretada por máquinas de manera automática, se necesita un lenguaje que ofrezca más posibilidades para la representación de términos y de las relaciones entre ellos.

Esto es precisamente lo que aporta OWL a RDF: la semántica necesaria para alcanzar los objetivos la nueva web.

Actualmente existen tres sub-lenguajes de OWL con distintos grados de complejidad para adaptarse a las necesidades de los usuarios [8]:

- *OWL Lite*: es el más sencillo y menos utilizado de las tres versiones. Sólo engloba algunos de los rasgos que proporciona el leguaje y limita el uso que puede hacerse de ellos.
- *OWL DL*: incluye los mismos conceptos que su versión completa con la diferencia de que sólo pueden aplicarse con ciertas restricciones. Por ejemplo, una clase puede ser subclase de muchas otras clases, pero no se permite que una clase sea instancia de otra clase.
- *OWL Full*: esta versión ofrece total libertad para la creación de ontologías, sin aplicar ningún tipo de restricción.

Cada uno de estos conjuntos es una extensión de su predecesor más simple, por lo tanto, una ontología expresada en OWL Lite será válida en OWL DL, aunque no a la inversa. Lo mismo ocurre para OWL DL y OWL Full.

En cuanto a la relación entre estos tres sub lenguajes y RDF puede afirmarse que un documento OWL, ya sea Lite, DL o Full, será a su vez un documento RDF. Sin embargo, la aserción contraria no puede hacerse debido a que el documento RDF puede no satisfacer las restricciones que tanto OWL Lite como OWL DL exigen. Las limitaciones que imponen estos dos tipos deben tenerse en cuenta al migrar un fichero RDF a OWL.

A continuación van a presentarse algunos de los términos del vocabulario de este lenguaje, dividiéndolos entre clases y propiedades.

2.3.1 Clases

owl:Thing

Es la clase más general en OWL. Todas las demás clases son subclases de ésta, y todos los individuos se consideran instancias de esta clase.

El uso de esta clase es similar a la superclase *Object* en algunos lenguajes de programación orientada a objetos.

owl:Class

Es el tipo que engloba los recursos RDF definidos como clase. Cuando se define una clase, esta se considera una instancia de la clase owl:Class.

Una clase representa ciertos individuos con las mismas propiedades.

owl:DataTypeProperty

Es la clase a la que pertenecen aquellas propiedades que tiene como rango el valor de un literal.

owl:ObjectProperty

Clase a la que pertenecen las propiedades que tienen como rango una instancia de owl:Class.

2.3.2 Propiedades

Algunas propiedades importantes:

rdf:type

Esta propiedad especifica que un recurso es una instancia de una determinada clase.

rdfs:subClassOf

Permite establecer jerarquías entre clases.

Especifica que una clase es subclase de otra concreta y por lo tanto todas las instancias de esta subclase serán también instancias de la superclase.

rdf:domain

Permite especificar que la propiedad tiene asociado un dominio de una clase concreta.

rdf:range

Propiedad que permite establecer el rango de una propiedad.
Como se vio con anterioridad, el rango de una propiedad limita los individuos que la propiedad puede tomar como valor.

Cabe destacar también el concepto de ***Individual,*** que hace referencia a las instancias que se crean de una clase.

Las propiedades de una ontología se utilizan para relacionar uno o varios individuos con otros.

2.3 Ontologías de sensores y SSN

A principios de 2009 se forma el *Semantic Sensor Networks Incubator Group* [1], SSN-XG, perteneciente al *World Wide Web Consortium*, W3C, con el fin de elaborar un modelo ontológico para la descripción de sensores y de las redes que éstos constituyen.
La ontología SSN, escrita en lenguaje OWL, surge a partir del trabajo desarrollado por este grupo, y es presentada en su última versión a mediados de 2011.

[1] http://www.w3.org/2005/Incubator/ssn/

Además de SSN, existen otras ontologías de sensores que fueron tomadas en consideración para el desarrollo de la que ocupa este PFC. [9]
A continuación van a presentarse algunas de estas ontologías, enfocadas fundamentalmente en la descripción de los sensores, en detrimento de las mediciones que estos realizan. [10]

- CSIRO Ontology: fue desarrollada como una ontología de sensores genérica para describir este tipo de dispositivos y sus despliegues. Abarca un amplio rango de conceptos relacionados con los sensores y a pesar de ser considerada una ontología basada en dispositivo, cubre también muchos aspectos de sus mediciones. Fue tomada como punto de partida y primera versión de SSN.

- OntoSensor: fue concebida para construir una base de conocimiento sobre sensores. Se centra especialmente en las especificaciones técnicas de los dispositivos, pero también permite describir las características de sus observaciones. La estructura de sus conceptos y propiedades es bastante desorganizada, por lo que no se recomendó para el desarrollo de SSN. [11]

- SWAMO (Sensor Web for Autonomous Mission Operations): Esta ontología fue diseñada para el permitir el intercambio y la utilización de información entre productos y servicios de la red de sensores de forma dinámica.

- SDO (Sensor Data Ontology): diseñada con el fin de poder realizar búsquedas relevantes dentro de redes de sensores heterogéneas y distribuidas.

- MMI Ontology (Marine Metadata Interoperability): ontología de dispositivos oceanográficos entre los que se incluyen sensores. Está muy enfocada en los componentes de un sistema y en el modo en el que

se organizan. Su principal prioridad es caracterizar los dispositivos de forma que un usuario (que bien podría ser una aplicación web) encuentre fácilmente aquellos datos o sensores que sean de su interés.

En la Figura 2-3 se recoge la capacidad de las ontologías mencionadas para la descripción de determinados aspectos relacionados con sensores. [12]

		SSN	CSIRO	OntoSensor	SWAMO	SDO	MMI
SENSOR	Sensor	*	*	*	*	*	*
	Tipo de sensor	*	*	*		*	
	Componentes	*	*	*	*		*
	Despliegue	*	*				*
	Configuración	*	*	*		*	*
	Acción y Proceso	*	*	*	*		*
OBSERVACION	Observación	*	*	*		*	
	Modelo de respuesta	*	*	*		*	
	Medida	*	*	*		*	
	Precisión	*	*	*		*	*
	Frecuencia	*	*	*		*	*
	Campo de visión/detección	*	*	*	*	*	
DATOS	Datos	*	*	*		*	
	Flujo de datos	*					
	Política de adquisicion de datos	*	*	*		*	*
	Dominio	*	*	*	*	*	*

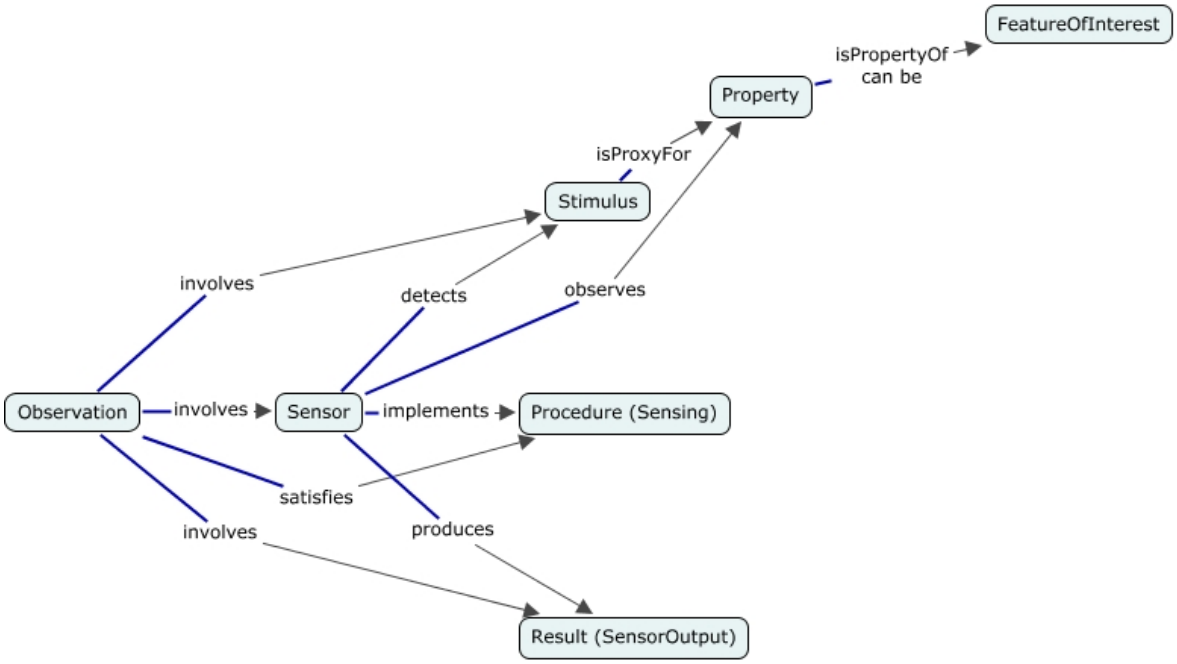
Figura 2-3: Comparativa entre ontologías de sensores

Esta comparativa permite comprobar que SSN es la ontología de sensores más completa. A su vez, es la que más relevancia y aceptación está teniendo en los últimos años. El hecho de que algunas de sus clases se encuentren alineadas con las de otra ontología de carácter más general, facilita su uso e interoperabilidad.

2.3.1 **Estructura de SSN**

El diseño principal de SSN está construido a partir de un modelo conocido como *Stimulus-Sensor-Observation*, SSO, en el que se describen y relacionan los dispositivos y los datos que estos recogen.

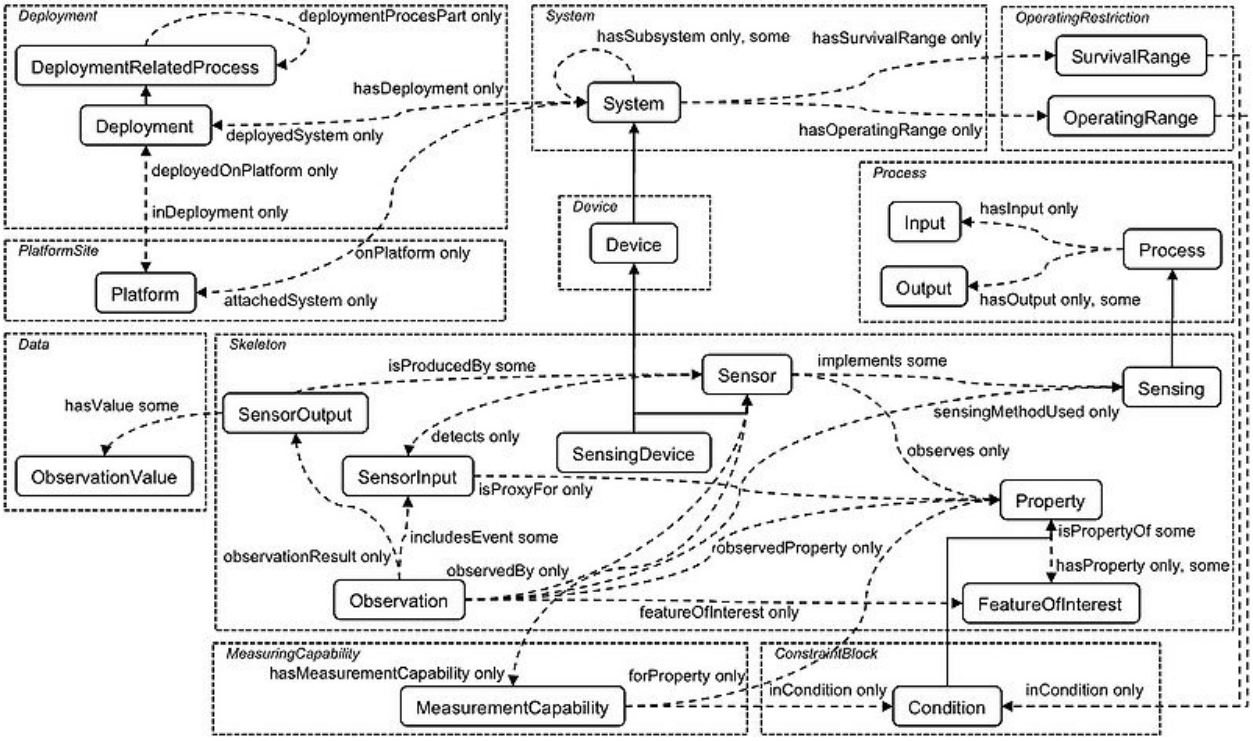
Esto permite realizar un uso de la ontología desde distintas perspectivas, ya sea centrándose en el propio sensor, las observaciones que realiza o el sistema en todo su conjunto.



Fuente: http://www.w3.org/2005/Incubator/ssn/wiki/File:SSO-pattern.jpg

Figura 2-4: Diseño Stimulus-Sensor-Observation

En la Figura 2-5 se pueden ver algunas de las clases más importantes de la ontología y las propiedades que las relacionan entre sí, dentro de los distintos módulos que la forman.



Fuente: http://www.w3.org/2005/Incubator/ssn/wiki/File:OntStructure-Overview.jpg

Figura 2-5: Visión general de SSN

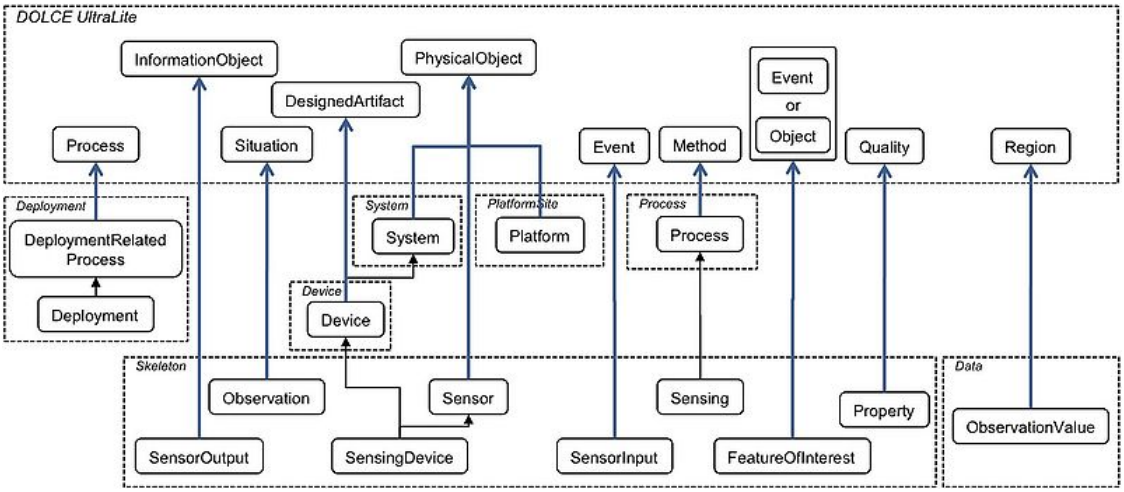
Algunas de las clases principales de SSN, y algunas propiedades, derivan de otras más generales de la ontología DOLCE Ultra Lite (DUL), evitándose así la necesidad de que redefinir conceptos ya contemplados en otra ontología.

La ontología DUL es la versión simplificada de la combinación de dos ontologías de nivel superior o fundamentales (o *upper ontology*, en inglés): DOLCE (Descriptive Ontology for Linguistic and Cognitive Engineering) y DnS (Descriptions and Situations).

Este tipo de ontologías se limitan a describir conceptos genéricos y suficientemente generales como para abordar una amplia gama de áreas de conocimiento. Según [13] ,

este tipo de ontologías son una pieza clave para la integración de conocimiento procedente de distintas fuentes.

En la Figura 2-6 pueden verse las clases mencionadas.



Fuente: http://www.w3.org/2005/Incubator/ssn/wiki/File:OntStructure-AlignmentDUL.jpg

Figura 2-6: Relación entre clases de SSN y DUL

2.3.2 **Clases y propiedades SSN**

La finalidad de este apartado es describir de forma breve los conceptos que modelan algunas clases y propiedades de la ontología, organizadas según los módulos que la conforman, y la clase SSN que las representa.

Únicamente se mencionarán las clases que se consideren más importantes.

2.3.2.1 Skeleton

Este módulo abarca las clases y propiedades necesarias para definir las características técnicas de un dispositivo de tipo de sensor: qué fenómenos observa, cuales son sus parámetros o valores de entrada y salida, etc.

Sensors

Para la ontología los sensores son objetos que realizan una observación del entorno (detectan un estímulo), y la transforman en algo diferente.
La visión tan general que ofrece acerca de estos dispositivos permite realizar una descripción de éstos con el nivel de detalle que sea necesario en cada caso. Puede definirse un sensor como "cualquier cosa que observe" o se le pueden ir añadiendo características por medio de propiedades, como por ejemplo, su rango de operación, la precisión con la que opera o la plataforma física en la que se encuentra.
A pesar de que se ha definido un sensor como un objeto, no debe considerarse la palabra en su sentido estricto, ya que la ontología no los limita únicamente a componentes técnicos o hardware. Una simulación, o incluso una persona, podría ser modelada como un sensor dentro de SSN.

Sensor Input

Se consideran cambios en el entorno que pueden ser detectados por un sensor y utilizados para la medición de una propiedad. Son los estímulos que "disparan" el sensor.
Un ejemplo de estímulo podría ser la corriente generada por las aspas de un molino de viento al girar.

Sensor Output

Valor de salida que se obtiene como resultado de una medición del sensor.

Property [2]

Son los atributos que miden los sensores a través de estímulos.

Observation

Representa aquello que ha observado el sensor. SSN dispone de varias propiedades para relacionar las instancias de esta clase, con instancias de algunas de las clases ya comentadas:

- observedBy: se utilizar para relacionar la instancia de la clase *Observation* con el sensor que ha registrado el fenómeno.
- observationResult: para relacionarla con el valor de salida del sensor, es decir, con una instancia de la clase *SensorOutput.*
- hasValue: apunta hacia el valor directo de la medición, que debe ser una instancia de la clase *ObservationValue*.

Sensing

Representa el modo en el que el sensor obtiene los valores del fenómeno que mide.

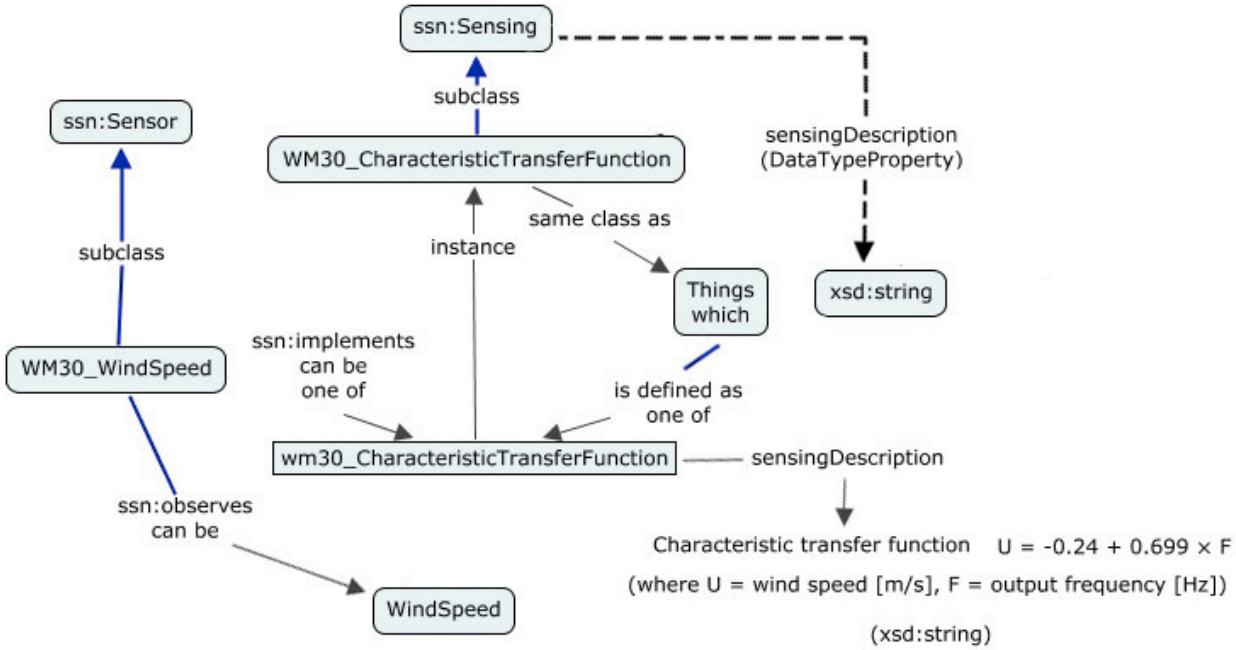
Siguiendo con el ejemplo anterior, podría utilizarse esta clase para definir la función de transferencia de un molino de viento, en la que se relaciona la frecuencia a la que giran las aspas con la velocidad del viento en ese momento.

(La información correspondiente a la función se expresa simplemente como una cadena de caracteres de tipo string).

$$U = -0.24 + 0.699 \times F$$

(where U = wind speed [m/s], F = output frequency [Hz])

[2] Es importante no confundir esta clase con el término "propiedad" utilizado para describir relaciones entre clases e individuos dentro de una ontología.

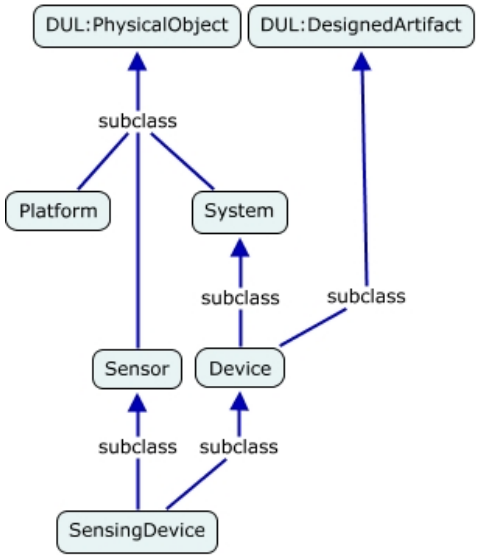


Fuente: http://www.w3.org/2005/Incubator/ssn/wiki/File:WM30_Method.jpg

Figura 2-7: Ejemplo de implementación de la clase Sensing

Sensing Device

Es la clase más utilizada para modelar un sensor. Es subclase de ssn:Device y ssn:Sensor. El hecho de que la clase Device sea a su vez subclase de ssn:System resulta de gran utilidad a la hora de describir nodos de una red formados por varios tipos de sensores, como se verá más adelante.



Fuente: http://www.w3.org/2005/Incubator/ssn/wiki/File:Ssn-Sensor_Sensing_SensingDevice_Device.jpg

Figura 2-8: Clase SensingDevice

La Figura 2-9 muestra las clases mencionadas, las relaciones entre ellas y de qué forma están alineadas con la ontología DUL. La clase *Stimulus* que aparece representada es equivalente a la clase *SensorInput* de la que se ha hablado más arriba.

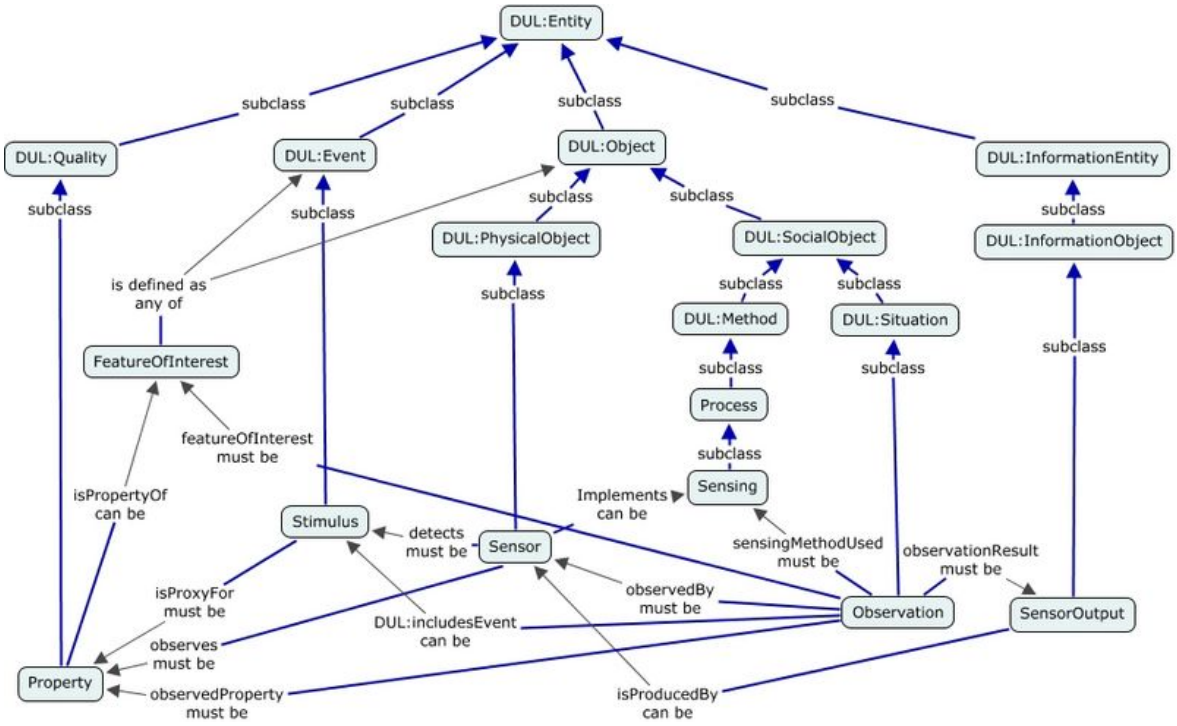
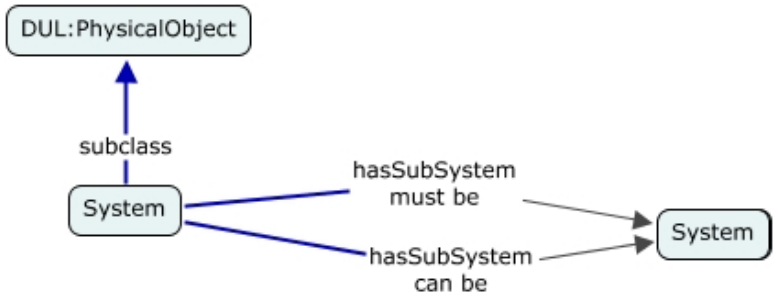


Figura 2-9: Implementación SSN del modelo SSO

2.3.2.2 System

Este módulo contempla una única clase, *ssn:System*, que puede utilizarse para representar toda la infraestructura de un despliegue de sensores.

En relación con esta clase, existe una propiedad, ssn:hasSubsystem, para indicar que un sistema puede estar formado por varios subsistemas más pequeños.



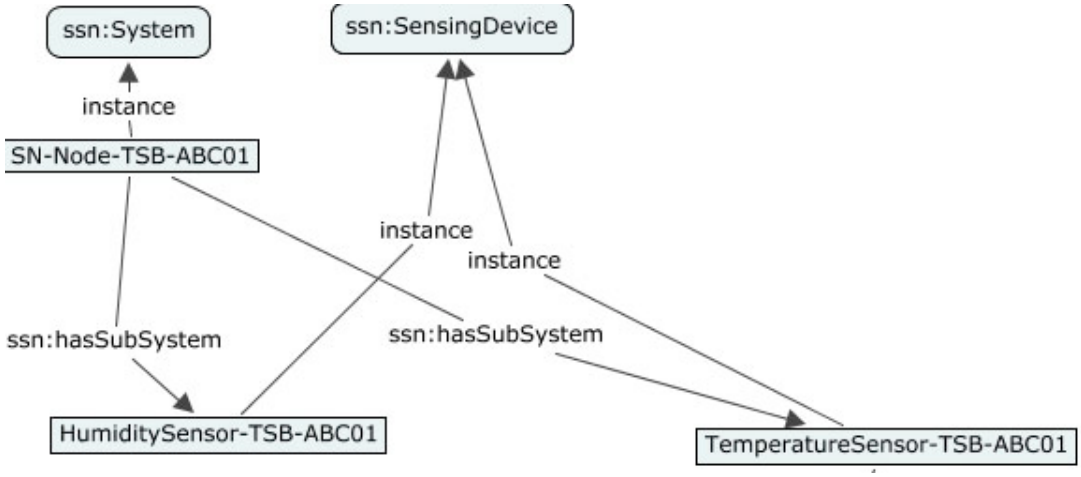
Fuente: http://www.w3.org/2005/Incubator/ssn/wiki/File:Ssn-System-Component.jpg

Figura 2-10: Clase *ssn:System*

Como se ilustra en la Figura 2-10, es requisito indispensable que aquello que se defina como subsistema sea a su vez un elemento de la clase *ssn:System* pudiendo dar a entender que no es posible modelar un sistema como un conjunto de sensores.

Sin embargo, como se ya se ha comentado, el hecho de que la clase *ssn:SensingDevice* herede de *ssn:System* se puede establecer que un sistema está compuesto de varios sensores.

A continuación se muestra un ejemplo en el que se implementa un nodo formado por un sensor de temperatura y otro de humedad.



Fuente: http://www.w3.org/2005/Incubator/ssn/wiki/File:SSN-UniSensors.jpg

Figura 2-11: Nodo formado por dos sensores

2.3.2.3 Measuring

Existe una propiedad muy importante dentro de este bloque y estrictamente relacionada con una clase con el mismo nombre para definir que un sensor posee unos atributos concretos en cuanto a las medidas que realiza y la forma que tiene de hacerlas y operar:

hasMeasurementCapability (propiedad)

MeasurementCapability (clase)

Haciendo uso de ellas, y de otras clases y propiedades adicionales, se pueden añadir infinidad de características y obtener una representación precisa de nuestro sensor.

En la Figura 2-12 se muestra de qué forma se relacionan las clases entre sí.

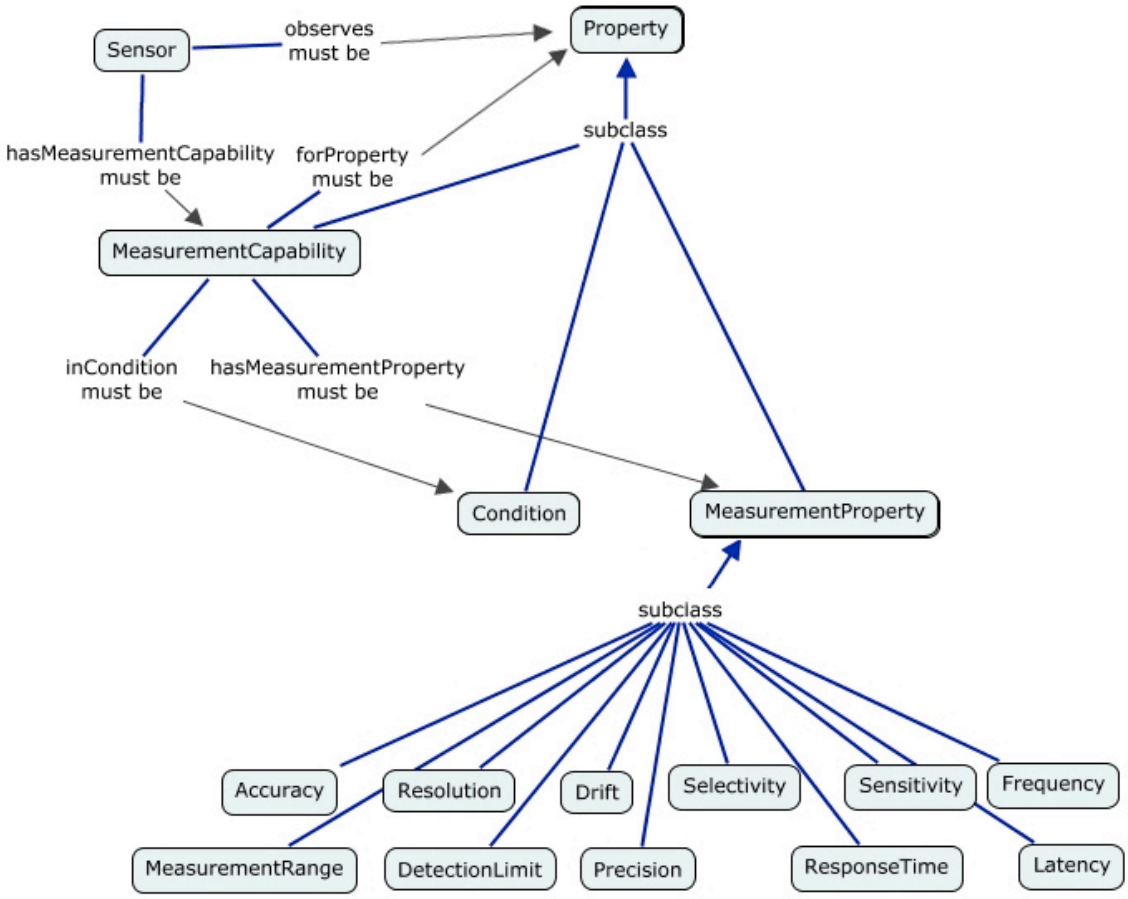
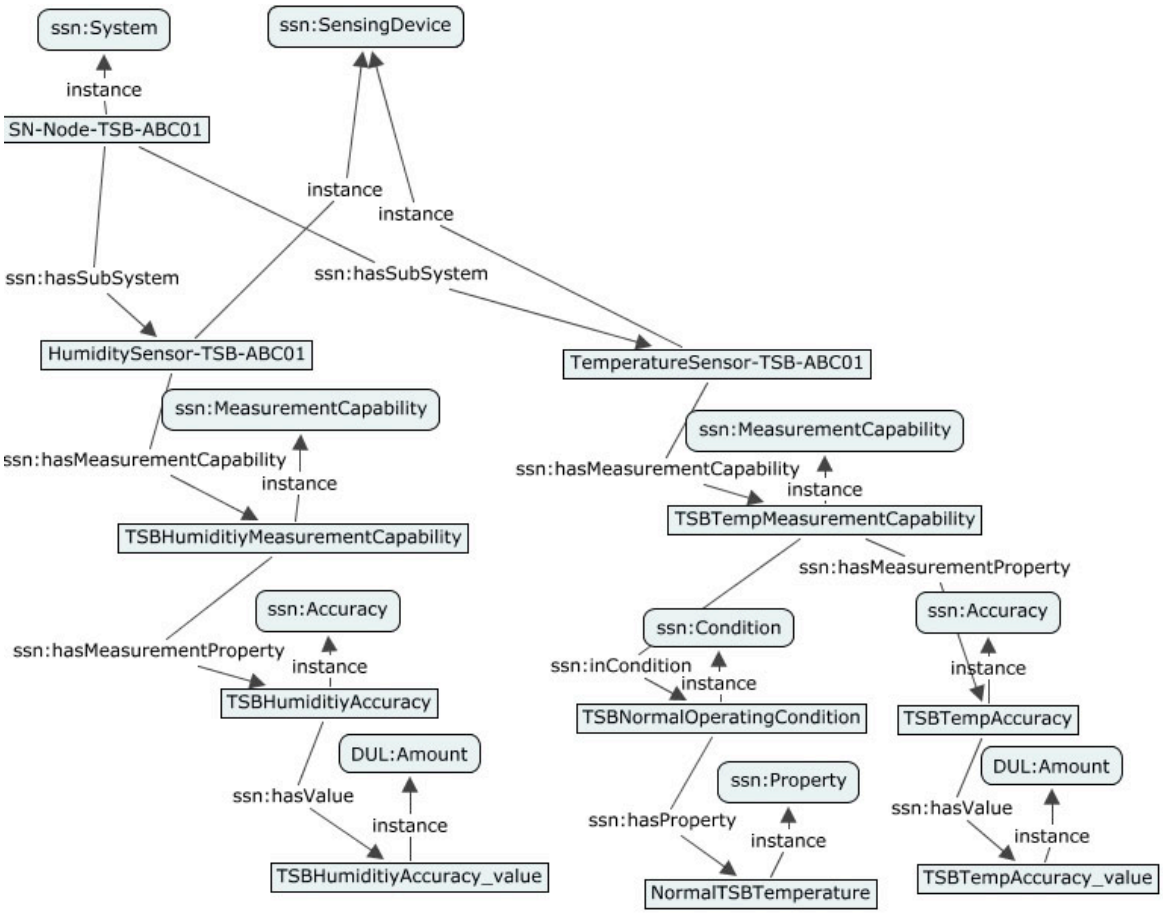


Figura 2-12: Capacidad de medición

En la siguiente ilustración, Figura 2-13, pueden observarse dichas clases aplicadas al ejemplo del nodo formado por dos sensores.



Fuente: http://www.w3.org/2005/Incubator/ssn/wiki/File:SSN-UniSensors.jpg

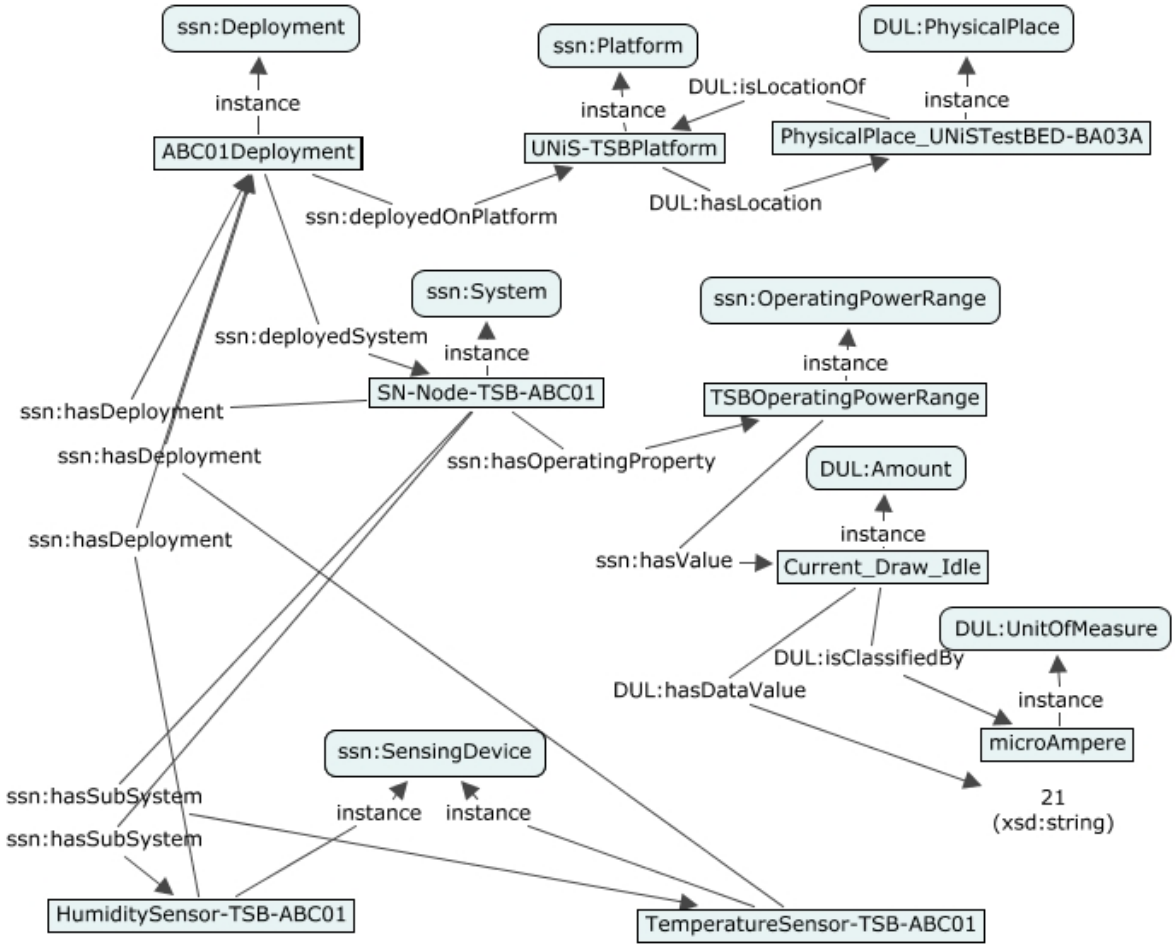
Figura 2-13: Ejemplo de uso MeasurementCapability y MeasurementProperty

2.3.2.4 Otros

Además de las clases y propiedades a los que se ha hecho referencia con anterioridad, la ontología cuenta con muchas otras con las que describir de forma exhaustiva una red o sistema de sensores.

En la Figura 2-14 se muestra parte de un despliegue en el que se hace uso de algunas clases no mencionadas hasta ahora. Como se puede ver, utilizando la clase *Platform* junto con propiedades de la ontología DUL en la que se apoya SSN, es posible definir el lugar físico en el que se encuentra el montaje.

También es posible indicar que el sistema opera dentro de unos límites de energía concretos, que requiere ciertas tareas de mantenimiento y un amplío etcétera de posibilidades.



Fuente: http://www.w3.org/2005/Incubator/ssn/wiki/File:SSN-UniDeploy.jpg

Figura 2-14: Ejemplo de despliegue

2.3.3 **Aplicaciones de SSN**

La aceptación que la ontología SSN está teniendo dentro de la comunidad de sensores está creciendo con rapidez y actualmente está siendo utilizada por numerosas organizaciones que van desde instituciones académicas y de investigación a organismos gubernamentales.

A continuación se presentan algunos de los proyectos que hacen uso de esta ontología.

2.3.3.1 SECURE (Semantics Empowered Rescue Environment) [14]

Este proyecto describe un sistema de tiempo real capaz de recoger datos producidos por dispositivos de rescate equipados con sensores de diversos tipos, y traducirlos a formato RDF utilizando la ontología SSN con el fin de que puedan ser procesados y utilizados para la detección de eventos en situaciones de emergencia.

Los robots de rescate permiten monitorizar el entorno sin poner en riesgo la vida de los primeros equipos de emergencia en acudir a una catástrofe, pero el flujo de datos que producen puede resultar contraproducente para la toma de decisiones si no se analizan de la manera adecuada. Es por esto que disponer de unos procedimientos que permitan interpretar la situación de manera precisa adquiera gran importancia.

Este sistema en concreto está enfocado en la detección de fuegos provocados por distintos tipos de combustibles, basándose en los datos recogidos por un sensor de temperatura, un sensor de infrarrojos, y varios sensores de detección de gases.

2.3.3.2 SPITFIRE: Semantic-Service Provisioning for the Internet of Things [15]

La ontología SSN se utiliza en este proyecto para describir las capacidades de los sensores, los sistemas y despliegues de los que forman parte, y las observaciones y mediciones que recogen los dispositivos. Además se han definido clases y propiedades extra con las que simplificar el desarrollo de servicios y aplicaciones basadas en datos recogidos por sensores. Se podría decir que igual que para el desarrollo de SSN se reutilizaron clases existentes en las ontología DUL, para la elaboración de la ontología SPITFIRE se ha hecho algunas de las clases proporcionadas por SSN.

Los esfuerzos de este proyecto se centran en unificar conceptos, métodos e infraestructuras software con las que facilitar el desarrollo de aplicaciones robustas y escalables para el Internet de las Cosas.

2.3.3.3 SemSorGrid4Env: Semantic Sensor Grids for Environmental Applications

El principal objetivo de este proyecto consiste en diseñar, implementar y desplegar una arquitectura orientada a servicios que permita desarrollar aplicaciones para la gestión medioambiental basadas en redes de sensores semánticas de gran escala. [16]

Las predicciones medioambientales resultan mucho más precisas si se combinan datos procedentes de sensores y fuentes distintas. Para la integración de los diferentes tipos de datos sea posible, se propone el uso de ontologías y de la semántica que estas proporcionan, siendo SSN la elegida para la descripción de las redes de sensores, sus dispositivos y las mediciones que se recogen de éstos.

2.3.3.4 Swiss Experiment Platform: Semantic Metadata and Querying [17]

SwissEx surge de la colaboración entre proyectos de investigación medioambiental y tecnológicos y tiene como finalidad principal la creación de una plataforma para el despliegue de redes de sensores que además permita manejar la información generada por los dispositivos.

Para la representación de la información relativa a los sensores y los datos obtenidos por estos se utiliza SSN.

3 Sistema de sensores

3.1 Modelo del sistema de sensores

Para la realización de este Proyecto de Fin de Carrera se ha definido un sistema de sensores compuesto por dispositivos de tres tipos diferentes: un sensor de temperatura, un dispositivo de luz, y un elemento resultado de la combinación de ambos que se ha denominado *Sensor Dummy*.

La elección de estos dispositivos y sus propiedades no ha sido arbitraria. Por un lado, se buscaba algo que no fuera complicado de describir semánticamente, pues el desarrollo de este trabajo ha supuesto la primera toma de contacto con un entorno ontológico de sensores. Por otro lado, el comportamiento de los terminales debía proporcionar al menos una operación de lectura y una de escritura, con y sin parámetros. La idea fundamental de esta última medida es que si se conseguían implementar satisfactoriamente estas operaciones, podría implementarse también el funcionamiento de dispositivos más complejos.

El hecho de que se haya definido el *Sensor Dummy* como un dispositivo formado por otros dos elementos, también es relevante. El caso que se ha contemplado en cuanto a las operaciones de funcionamiento, se puede aplicar también para el número de dispositivos que componen un terminal. El modo de representar e implementar el sencillo *Dummy* que se ha utilizado a modo de ejemplo para este trabajo, no debería diferir demasiado de los pasos a seguir si se tuviera una mota perteneciente a una red de sensores real.

A lo largo de este apartado se van a presentar la estructura y características de cada uno de estos componentes, cuya especificación, como se ha comentado anteriormente, se ha realizado haciendo uso de clases y propiedades correspondientes a la ontología SSN.

Con el fin de obtener una mejor comprensión de las figuras que se utilizarán para representar las características de los distintos elementos, se han establecido las siguientes consideraciones: Para diferenciar la ontología a la que pertenecen las clases y propiedades que se han utilizado, se ha añadido el prefijo "ssn:" y "DUL:" a sus nombres. Además, las clases de SSN se encuentran representadas por el color morado.
A las instancias se les ha dado el color azul. El nombre con el que están indicadas, es el nombre con el que han sido creados los *individuals*, exceptuando el caso del sensor de temperatura (mostrado como *TemperatureSensor*), el *Dummy* (*SensorDummy*), el dispositivo de luz (*LightDevice*) y los métodos java (*JavaMethod*). Estos tipos deben recibir un nombre diferente por cada instancia que se cree de sus correspondientes clases para poder ser diferenciados unos de otros dentro del modelo. Como se vio con anterioridad, los recursos de un documento se hace a través de sus identificadores URI, y deberán ser distintos para objetos distintos.
Para el resto de clases, puede utilizarse el mismo nombre para todas las instancias, pues no existe diferencia entre unas y otras.

3.1.1 **Sensor Dummy**

Los elementos de este tipo se han descrito como *individuals* de la clase SSN *System*. Como vimos en el apartado relativo al estudio de la ontología, esta clase permite definir dispositivos complejos formados por varios componentes mediante la propiedad *hasSubsystem*. En este caso se ha utilizado dicha propiedad para relacionar las instancias correspondientes al sensor de temperatura, modelado como un *individual* de la clase *Sensing Device* y el dispositivo de luz, creado a partir de la clase *Device*.

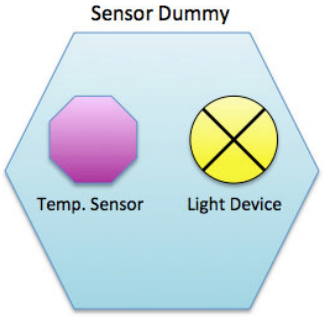


Figura 3-1: Sensor *Dummy*

Una característica importante a la hora de trabajar con sensores, y especialmente si se cuenta con un despliegue de más de un sensor, es la ubicación en la que se encuentra, ya que las lecturas de un sensor pueden ser insuficientes para la toma de decisiones si no se conoce el lugar en el que se están produciendo. Por este motivo se ha decidido añadir al modelo esta propiedad.

SSN no proporciona la forma de establecer la ubicación de un sensor, pero si lo hace la ontología DUL con la que está estrechamente relacionada.
Para este trabajo se ha propuesto como ubicación una cadena de caracteres que recoge las coordenadas GPS de la supuesta ubicación del sensor, pero podría darse únicamente un nombre del lugar o incluso definir una instancia de la clase *Physical Place* de la ontología DUL.

Puesto que el objetivo principal de este PFC es la incorporación del modelo de sensores en su formato OWL a una aplicación java, es necesario definir una propiedad que recoja los métodos que el sensor debe implementar, correspondientes a su funcionamiento. Para ello se hace uso de la propiedad de SSN *implements*, y de la clase *Method* de esta misma ontología.

Para estos dispositivos se han definido dos métodos:

- *listComponents*: que devolverá los nombres de los componentes que constituyen el *dummy*.
- *getLocation*: que devuelve ubicación del sensor.

Además, deberá implementar los métodos correspondientes al funcionamiento de los elementos que lo componen: el sensor de temperatura y el dispositivo de luz.

La Figura 3-2 representa la estructura del *Sensor Dummy* junto a sus propiedades. En la imagen se puede observar que los elementos se han definido como *individuals* de las clases SSN, tal y como se ha explicado con anterioridad.

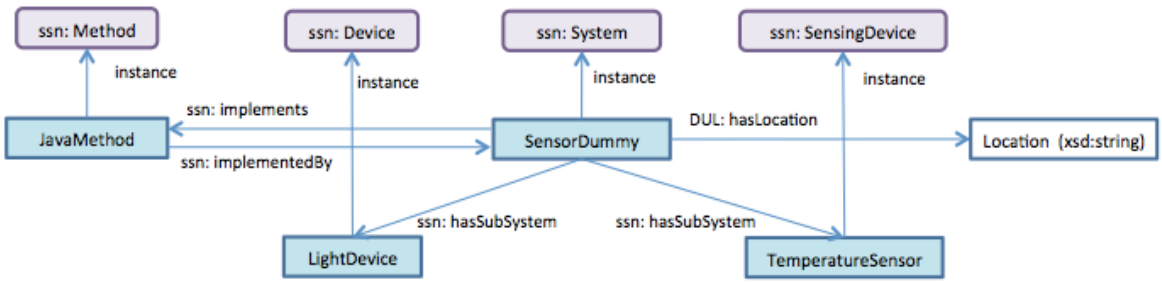


Figura 3-2: Estructura del *Sensor Dummy*

3.1.2 **Dispositivo de luz**

Los dispositivos que se han modelado cuentan con un indicador luminoso de intensidad regulable para valores entre 0 y 100. También recogen el estado del dispositivo, pudiendo éste ser "Apagado" o "Encendido".

El funcionamiento que se ha propuesto para este tipo de componentes es el siguiente: el estado del dispositivo se podrá controlar o bien de manera independiente mediante un supuesto interruptor de Encendido/Apagado, o en función de la luminosidad que se establezca para el dispositivo. Si se elige un valor de 0 para la intensidad, el estado del terminal pasará automáticamente a "Apagado".

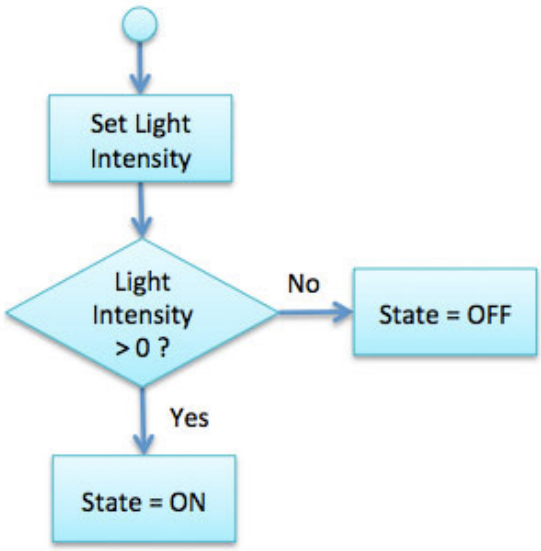


Figura 3-3: Funcionamiento del regulador de intensidad luminosa

Los métodos que se deberán implementar son:

- *getLightIntensity*: que devolverá el valor de intensidad que tenga asociado en ese momento el dispositivo.

- *setLightIntensity*: para poder regular la intensidad deseada en cada momento.
- *getState*: que devolverá si el dispositivo se encuentra apagado o encendido.
- *setState*: para apagar o encender el dispositivo.

Para modelar este comportamiento se ha hecho uso de la clase *Process*, que permite describir los procedimientos de funcionamiento de los dispositivos. Esta clase cuenta con dos propiedades básicas con los que relacionar los parámetros de entrada y salida, que se deberán modelar como instancias de las clases *Input* y *Output*.

Como ocurría con el *Dummy* es necesario incorporar al modelo del dispositivo los métodos Java que será capaz de implementar una vez se haya hecho el desarrollo de la aplicación.

Si bien la propiedad SSN *implements* la se ha utilizado para tal fin al relacionarla con un *individual* de la clase *Method,* en este caso, como puede verse en la Figura 3-4, se usa también para relacionar el dispositivo a las instancias que definen su comportamiento.

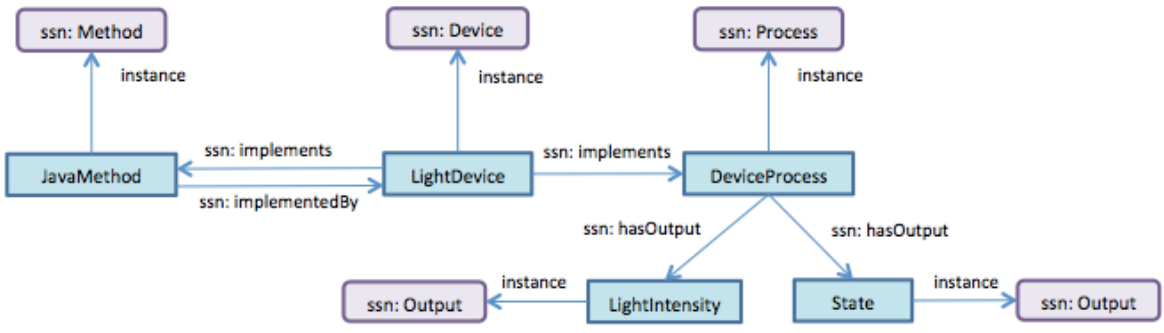


Figura 3-4: Dispositivo de luz

3.1.3 **Sensor de temperatura**

El sensor de temperatura es el elemento que más clases y propiedades utiliza de la ontología de SSN, ya que además de definir el propio sensor, es necesario caracterizar las mediciones que realiza.

Para modelar este dispositivo se han utilizado instancias de la clase *Sensing Device* de SSN. En un primer momento se contempló la posibilidad de definir una subclase de ésta con la que representar de forma general un sensor de temperatura y a partir de ella, ir creando los *individuals* que fueran necesarios.

La posibilidad de crear subclases resulta de gran utilidad en escenarios en los que se cuente con sensores de distintos tipos. Es decir, en un supuesto caso en el que se hubiera contado con sensores de aceleración, movimiento, o incluso teniendo únicamente sensores de temperatura pero de distintos fabricantes o modelos, la opción más apropiada habría sido la de definir una subclase "Sensor Temperatura" y crear los *individuals* a partir de ella. De este modo, todos los *individuals* creados quedarían definidos como sensores de temperatura, y se añadirían las propiedades complementarias necesarias para diferenciarlos entre sí.

Debido a que el modelo con el que se ha trabajado a lo largo de este PFC únicamente contempla un tipo de sensor, no ha sido necesario definir una subclase, y todos los sensores de temperatura que quieran representarse contarán con las mismas características.

Puesto que al crear los dispositivos como instancias de *Sensing Device* no se proporciona información sobre el tipo de sensor que representan, se utilizará la propiedad SSN *observes* para este fin. Esta propiedad relaciona una instancia de tipo sensor con una instancia de la clase *Property* de SSN, que en este caso se creará como temperatura.

La primera característica del sensor que se ha definido es su capacidad de medición. En la Figura 3-5 se pueden observar las clases que se han utilizado para ello, y las propiedades con las que se relacionan las distintas instancias.

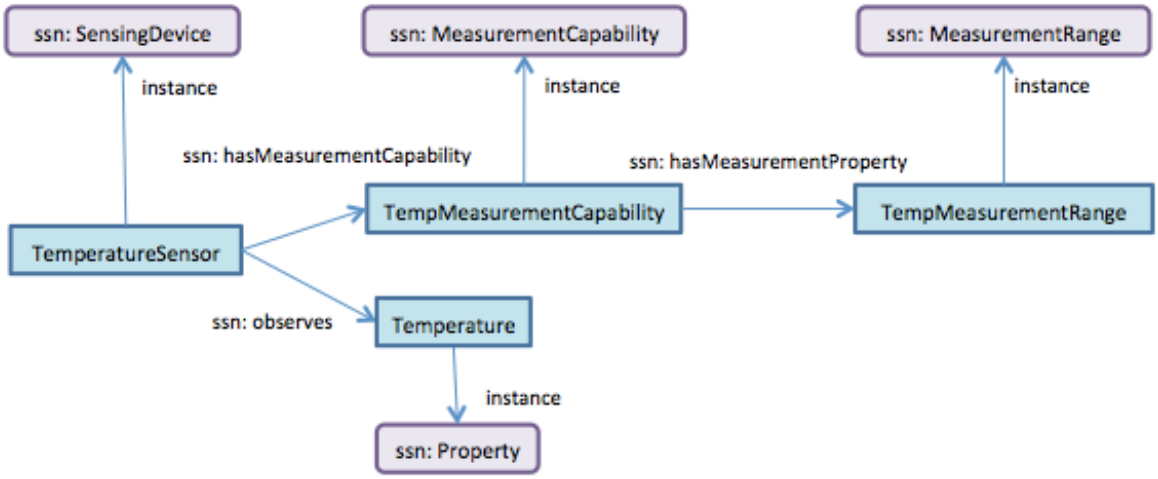


Figura 3-5: Capacidad de medición del sensor de temperatura

Cuando se presentó la ontología se vio que a partir de la capacidad de medición del dispositivo se pueden añadir infinidad de atributos al sensor modelado. Para este proyecto se estableció que el sensor estaría definido por un rango de operación, por lo que para su descripción semántica se ha escogido la propiedad SSN correspondiente a esta característica: *MeasurementRange.*

Utilizando esta clase únicamente se puede determinar que el sensor podrá realizar mediciones dentro de un rango, pero no es posible definir los límites de dicho rango. Para ello se han creado dos propiedades auxiliares, ajenas a la ontología SSN, con las que dar los valores correspondientes.
Puesto que el sensor realiza observaciones de temperatura, los valores máximo y mínimo que puede registrar deberán ser de este tipo.

La Figura 3-6 ilustra de forma detallada la especificación que se ha hecho del rango del sensor.

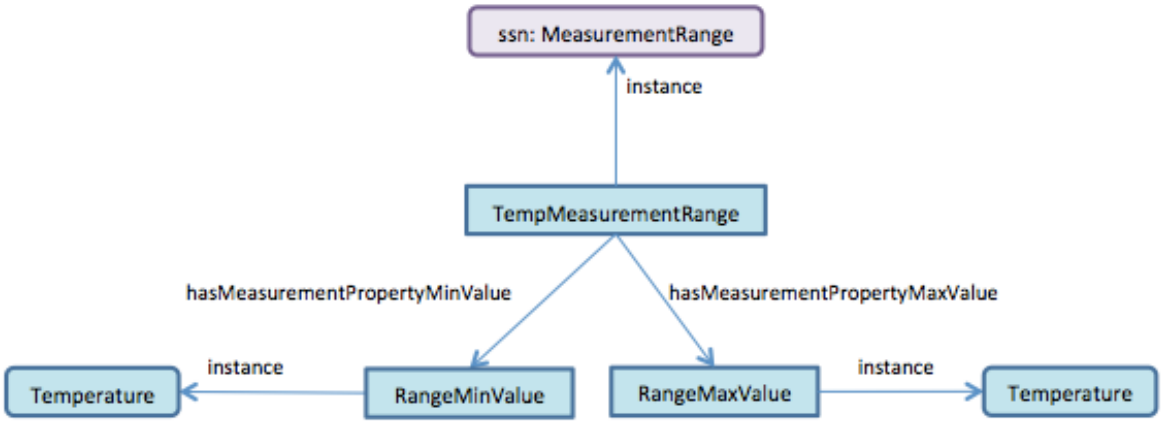


Figura 3-6: Rango de operación del sensor de temperatura

Una vez definidas las propiedades del sensor, se modelan los valores que este registra. La Figura 3-7 recoge las clases que proporciona SSN para ello y la forma en la que deben relacionarse unas con otras. Como se puede observar, la propiedad *observedBy* permite vincular las instancias de la clase *Observation* correspondientes a las mediciones del sensor con el dispositivo concreto que las realiza.

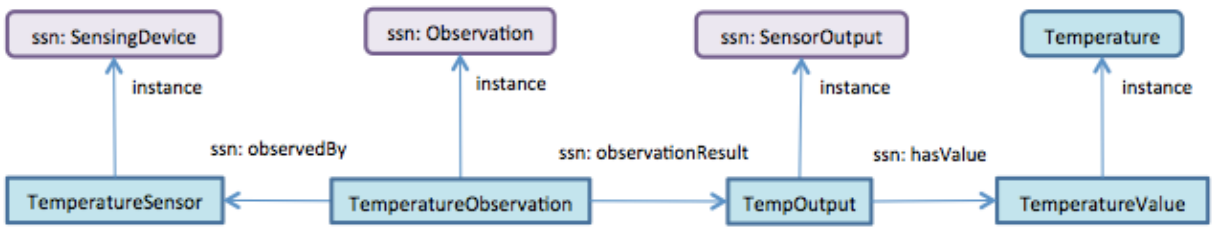


Figura 3-7: Mediciones del sensor de temperatura

Como se ha podido comprobar todos los elementos que se han modelado han sido creados como instancias de clases pertenecientes a la ontología SSN, a excepción de las mediciones de temperatura, que se han definido a partir de una clase propia.

Estos valores se podrían haber definido como instancias de la clase *ObservationValue* que proporciona SSN, pero se decidió crear una subclase de ésta a pesar de no ser realmente necesario. El motivo no es otro que el de utilizar esta propiedad que proporcionan las ontologías.

La Figura 3-8 muestra cómo la clase *Temperature* hace uso de las dos ontologías a las que se ha hecho referencia en esta memoria para definir que los valores de temperatura modelados tendrán como unidades de medida grados Celsius y su valor numérico será de tipo *float*.

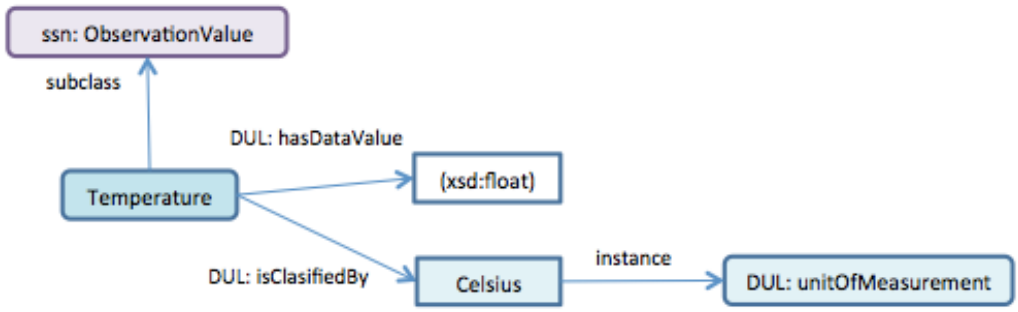


Figura 3-8: Clase *Temperature*

4 Diseño e implementación

A lo largo de este capítulo va a detallarse tanto el diseño de la aplicación global que se ha desarrollado, como el de los diferentes elementos que la componen. Inicialmente se hará mención a una de las tecnologías empleadas para la implementación de los mismos.

4.1 Jena

Durante esta fase del PFC, para la implementación de los diversos programas que se irán desarrollando, se utilizará Jena, una API de Java especialmente diseñada para el desarrollo de aplicaciones semánticas.

Este *framework* es capaz de trasladar a lenguaje Java las diferentes clases, propiedades e *individuals* que haya definidas en una ontología, lo que permite añadir nuevos elementos a un modelo concreto y manejar los ya existentes de manera sencilla.

El entorno que ofrece Jena no es dependiente del lenguaje o formato que se utilice para la representación de la ontología. Las clases Java que proporciona para la representación de las clases de la ontología, *OntClass*, serán las mismas independientemente de si se trata de una clase OWL o una clase RDFS [18]. Las diferencias entre las diversas representaciones existentes se reflejan al crear el modelo abstracto, *OntModel*, en el que Jena almacena la ontología por medio del perfil que se pase por parámetro al método de creación de dicho modelo.

Al trabajar con una ontología en Jena no se modifica su representación RDF, simplemente se proporcionan unos métodos y clases adicionales para con las que manejar las tripletas RDF de la ontología de forma sencilla. Es importante señalar que los objetos *OntClass* no almacenan información propia de la clase ontológica que representan y por lo tanto, al ejecutar por ejemplo un método para listar las propiedades de dicha clase, Jena tendrá que consultar el modelo y buscar las tripletas RDF correspondientes a la información buscada.

De igual forma, al definir una subclase no queda asociada al objeto *OntClass* a partir del cual se describa, sino que directamente se añade una nueva tripleta al modelo utilizando como predicado la propiedad "rdfs:subClassOf" y como sujeto y objeto, la clase y la subclase correspondientes.

Esta herramienta tiene un papel fundamental dentro del desarrollo de las aplicaciones gracias a la gran cantidad de funciones para el manejo de modelos OWL y ontologías que proporciona.

4.2 Diseño general de la aplicación

En el capítulo anterior se hizo mención al sistema de sensores que la aplicación final se encargará de representar de forma automática. Este sistema debe presentarse como un fichero de tipo OWL en el que todos sus dispositivos estén definidos de acuerdo a las especificaciones que se han establecido.

Describir los elementos uno a uno puede resultar tedioso, especialmente si se quiere modelar un sistema extenso, ya que sería necesario definir numerosos *individuals* y propiedades para cada dispositivo.

Con el fin de simplificar esta tarea se ha diseñado un pequeño programa Java que permite crear tantos dispositivos como sea necesario para modelar el sistema de sensores utilizando las instancias y propiedades SSN correspondientes, y una vez esté completo, escribirlo en un fichero OWL.

A partir de este documento, se generarán las interfaces Java correspondientes a cada tipo de dispositivo, en las que aparecerán indicados los métodos que se definieron al crear las descripciones semánticas de los componentes del sistema.

Estas interfaces se implementarán de acuerdo al funcionamiento establecido para los distintos terminales y se integrarán con las clases adicionales que sean necesarias para el desarrollo de la aplicación final.

El esquema de la
Figura 4-1 representa los distintos bloques de los que se ha hablado en este apartado y que componen el desarrollo global del PFC.

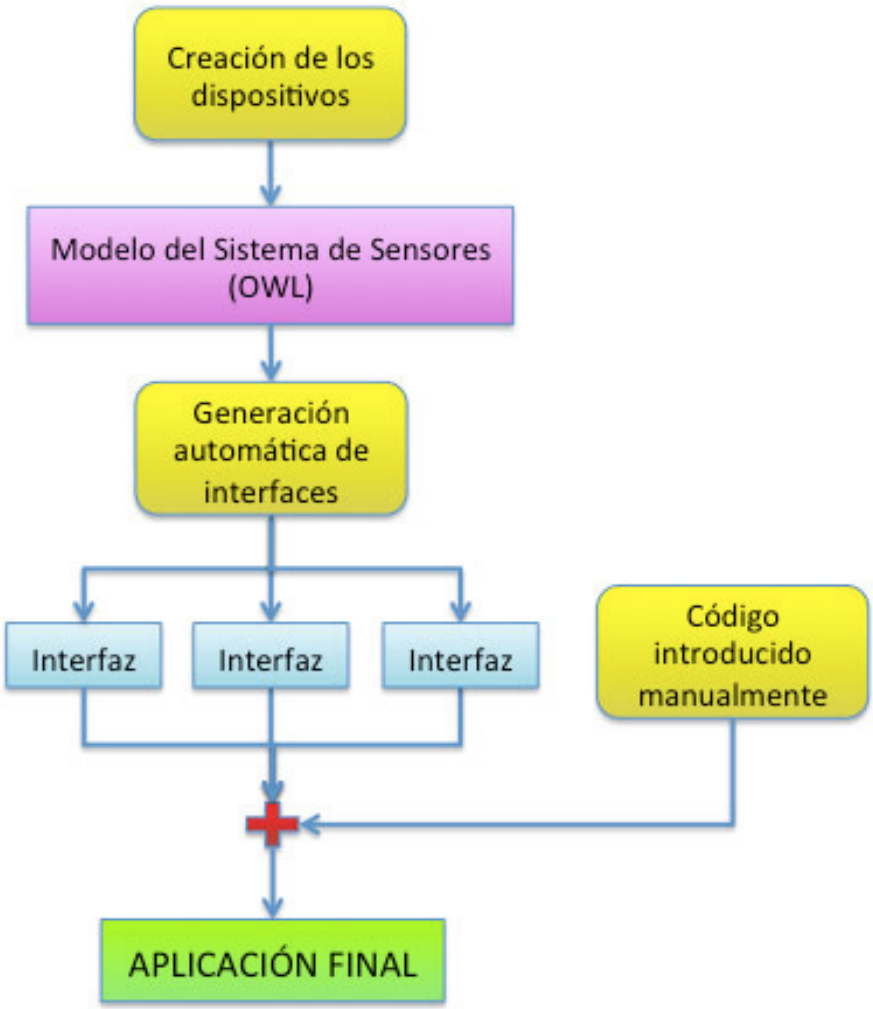


Figura 4-1: Diseño general

4.3 Detalle e implementación

En esta sección se va a tratar en detalle la forma en la que se han implementado cada uno de los bloques que componen el sistema presentado en el apartado anterior.

4.3.1 **Creación de los dispositivos**

Para desarrollar el programa de creación de *individuals* de los distintos elementos del sistema de sensores se ha hecho uso de las funcionalidades que Jena proporciona para el manejo de ontologías.

Como primer paso es necesario definir una variable de tipo *OntModel* para almacenar los ficheros OWL correspondientes a la ontología SSN y DUL. Las instancias que se vayan creando a continuación se incorporarán a este modelo por lo que cuando se exporte como fichero OWL, el documento contendrá las ontologías utilizadas además de los elementos del sistema de sensores que se hayan descrito.

En la Figura 4-2 se muestra el bloque correspondiente a la aplicación que se está detallando.

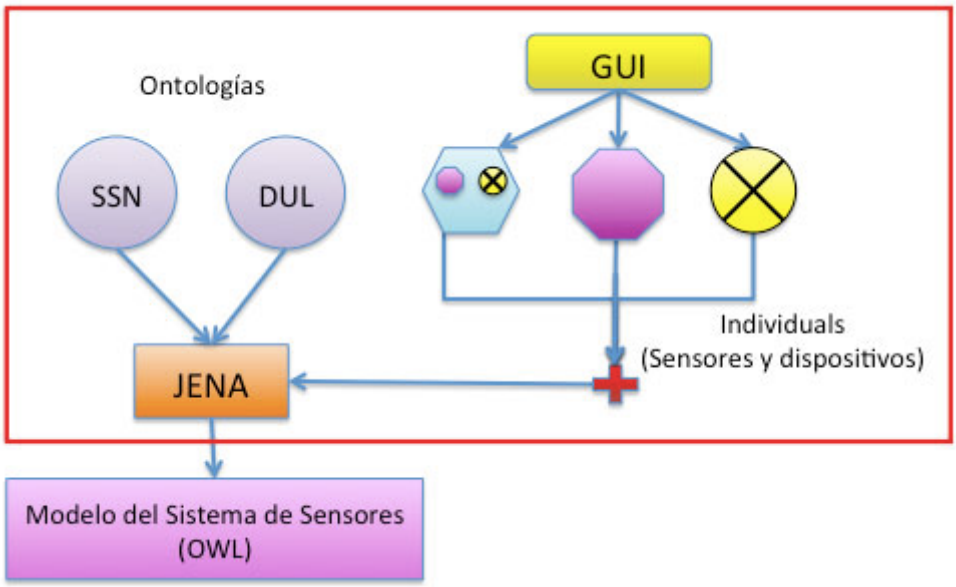


Figura 4-2: Creación de dispositivos

La interfaz gráfica de la que consta la aplicación permite la creación de un elemento proporcionando únicamente el nombre que se le quiera dar en el caso de los sensores

de temperatura y los dispositivos luminosos, y el nombre y una ubicación en el caso de los *Dummy*.

El programa se encarga de crear los *individuals* necesarios para cada terminal y de relacionarlos entre sí de la manera adecuada gracias a la implementación de unos métodos que reciben como parámetro el modelo Jena en el que deben crearse y las variables de tipo string con la información relativa a los nombres y ubicación deseadas.

Existe la posibilidad de crear los sensores de temperatura y los dispositivos de luz de manera independiente, pero no se contempla la creación de un *Dummy* sin indicar los dos elementos que deben formar parte de él, puesto que de acuerdo a la definición que se ha hecho de estos dispositivos, no pueden existir sin sus dos componentes. Cabe destacar que si existe un sensor de temperatura y un dispositivo de luz creados previamente, para la construcción de un nuevo *dummy* se podrán utilizar estos elementos sin que tengan que crearse dispositivos adicionales.

La Figura 4-3 y la Figura 4-4 reflejan la parte de la interfaz gráfica con la que crear los dispositivos correspondientes de la forma que se ha comentado.

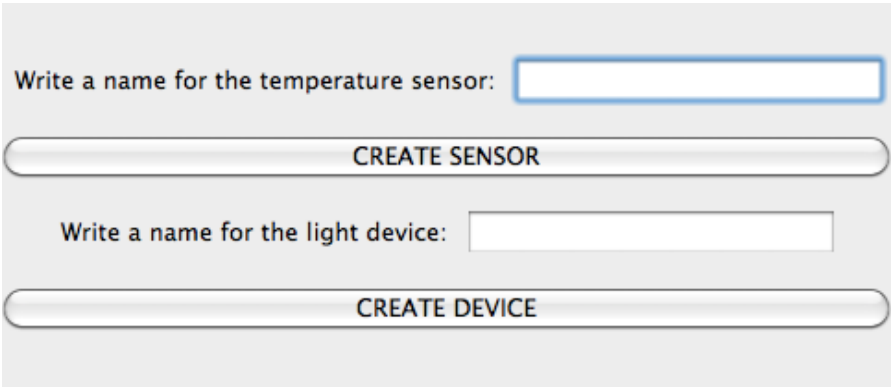


Figura 4-3: GUI para la creación de sensores y dispositivos de luz

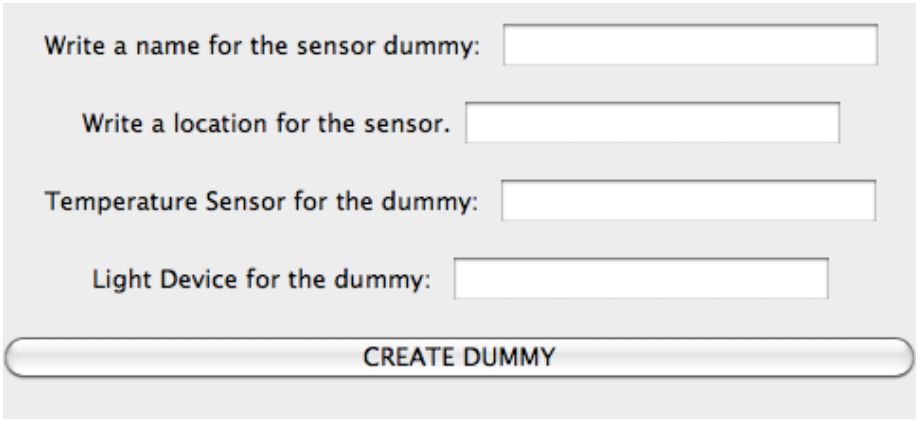


Figura 4-4: GUI para la creación de un *Dummy*

Para conocer en todo momento el estado del sistema que se está modelando, la aplicación cuenta con un espacio en el que se listan de manera automática los elementos a medida que se van creando y permite consultar las propiedades de cada uno de ellos.

Una vez creados los componentes necesarios, es suficiente proporcionar un nombre para generar el fichero OWL en el que se escribirá el sistema.

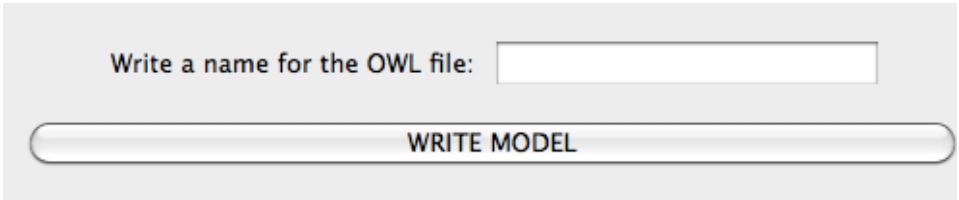


Figura 4-5: Escritura del fichero OWL con el modelo del sistema

4.3.2 **Generación automática de interfaces.**

A partir del fichero OWL que se generó en el paso anterior es posible crear una interfaz Java por cada tipo de dispositivo del sistema. Puesto que todos los elementos del mismo tipo se comportan de igual manera y por lo tanto deberán implementar los mismos métodos, no es necesario generar una interfaz por cada dispositivo que se haya descrito.

A pesar de que se han definido tres tipos de dispositivos distintos, y por lo tanto tres clases de interfaces diferentes, podría darse un sistema que únicamente contara con uno o dos tipos de terminales (sensores de temperatura y/o dispositivos de luz). En tal caso, solo se generarían las interfaces correspondientes a los elementos modelados en el sistema, ya que de las demás no se tendría constancia.

La Figura 4-6 representa la interfaz java generada automáticamente para los dispositivos de *tipo Dummy*, Como se puede observar, los elementos que componen el *dummy* están referenciados en la interfaz, y sus métodos deberán ser implementados junto a los métodos propios del *dummy*.

```
import com.hp.hpl.jena.ontology.OntModel;

public interface IDummy extends ITemperatureSensor, ILightDevice {
    public String[] listComponents(OntModel model, String dummyName);
    public String getLocation(OntModel model, String dummyName);
}
```

Figura 4-6: Interfaz del *Sensor Dummy*

Esto es debido a que la generación de las interfaces se realiza recorriendo el fichero OWL en el que está modelado el sistema de sensores en busca de los *individuals* de las clases a partir de las cuales se hayan definido los dispositivos, en este caso aquellos que sean de tipo *System*, *Sensing Device* y *Device*.

Para poder trabajar con el modelo de sensores se hace uso de la misma variable Jena en la que se almacenaban las ontologías DUL y SSN, *OntModel*, pero cargando en esta ocasión el documento correspondiente a la representación del sistema.

```
OntModel model = ModelFactory.createOntologyModel(OntModelSpec.OWL_MEM);
InputStream entrada = new FileInputStream("systemModel.owl");
model.read(entrada, null);
entrada.close();
```

Figura 4-7: Carga del modelo de sensores en Jena

La librería proporciona también un iterador con el que se pueden listar todos *individuals* presentes en el modelo, y para cada uno de ellos enumerar sus propiedades.

Estas propiedades son las que se definieron para cada una de las instancias en el momento de la creación de los dispositivos, y las que se deberán tener en cuenta para la búsqueda de las características de cada elemento.

```
/* Se recorre el modelo buscando individuals */
ExtendedIterator<Individual> i = model.listIndividuals();
      while (i.hasNext()) {
            Individual indi = i.next();
            /* Se listan las propiedades de los individuals */
            for (StmtIterator j = indi.listProperties() ; j.hasNext(); ) {
                  Statement s = j.next();
                                    [...]
            }
      }
```

Figura 4-8: Listado de *individuals* y propiedades

Cuando la aplicación encuentra una instancia del tipo buscado, debe localizar la propiedad con la que se relacionaron los métodos Java que debe implementar y con ellos construye la interfaz correspondiente.

En la interfaz de las instancias de tipo *System* con las que se han modelado los *Dummy,* se debe hacer referencia a los dos elementos que lo componen, ya que a la hora de realizar la implementación, se deberán desarrollar también los métodos del sensor y del dispositivo de luz.

4.3.3 **Implementación de la aplicación final**

Llegados a este punto se tienen todos los elementos necesarios para pasar al desarrollo de la aplicación final correspondiente a este proyecto de fin de carrera, con la que se podrán representar gráficamente los dispositivos modelados en un fichero OWL.

Los documentos con los que se ha trabajado para llevar a cabo esta implementación se han generado por medio de la aplicación para la creación de modelos que se ha descrito con anterioridad, pero también podría utilizarse un fichero OWL generado por otras vías siempre y cuando los dispositivos representados estuvieran correctamente definidos de acuerdo a las estructuras y características que propone la ontología SSN.

La implementación de las interfaces y sus métodos se ha elaborado conforme al funcionamiento estipulado para cada uno de los elementos, por lo que ha sido posible simular el comportamiento de los dispositivos en vez de plasmar únicamente su representación gráfica.

Puesto que el desarrollo de esta aplicación también se ha realizado en Java, ha sido necesario el uso de la librería Jena para el manejo del modelo de sensores. Como ocurría para los casos anteriores, el fichero OWL correspondiente al sistema se ha asociado a una variable *OntModel* a raíz de la cual se ha ido trabajando con los *individuals*.

La representación del sistema y de cada uno de los distintos elementos que lo componen se ha hecho utilizando AWT, un kit de herramientas de Java para la creación de gráficos e interfaces de usuario. Este paquete proporciona la clase *Panel* a partir de la cual se han modelado los dispositivos y la clase *Marco*, que se ha empleado para reproducir la totalidad del sistema de sensores.

Para cada tipo de componente se ha definido una clase java, que extiende la mencionada *Panel*, a la que se han añadido diversos elementos gráficos en función de los requerimientos de cada uno. (Botones, cuadros de texto, etc.)

La representación gráfica del sistema y sus componentes se construye a partir de la variable *OntModel* ligada al fichero OWL con el modelo de sensores. Este modelo debe recorrerse utilizando procedimientos de búsqueda de *individuals* similares a los que se utilizaron para la generación de las interfaces. Si bien en ese caso sólo era necesario producir una interfaz por cada tipo de instancia buscada, para esta situación se deben crear tantos objetos de sus clases java equivalentes como dispositivos se encuentren descritos en el modelo. Debe tenerse en cuenta que en esta ocasión, aunque se disponga de varios dispositivos iguales, es necesario representarlos todos.

La Figura 4-9 ilustra la representación gráfica de un sistema de sensores compuesto por un único sensor Dummy. Los componentes que forman parte de este dispositivo (TemperatureSensor1 y LightDevice1) se encuentran dibujados de manera independiente para que sea más fácil observar su funcionamiento.

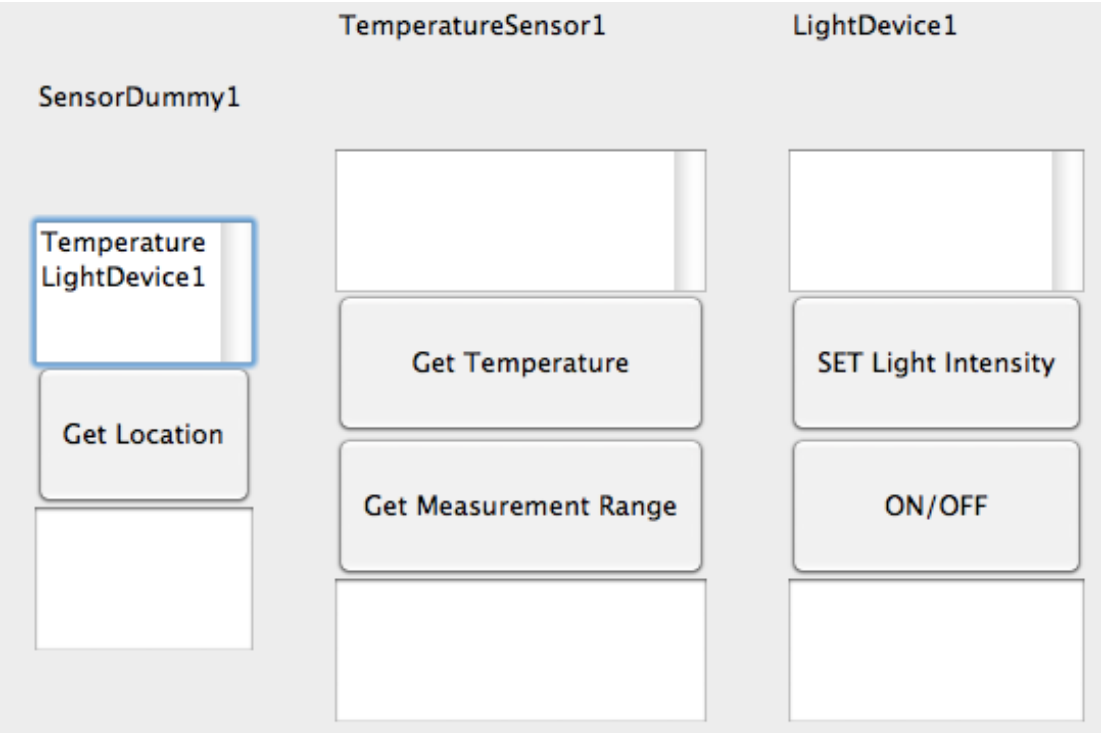


Figura 4-9: Sistema compuesto por un único *Dummy*

5 Conclusiones y trabajo futuro

Para finalizar, este capítulo recoge las conclusiones que se pueden obtener del trabajo desarrollado a lo largo de este proyecto de fin de carrera, así como un pequeño análisis de las diferentes secciones que han compuesto esta memoria.
Así mismo, se realizará una revisión de los objetivos planteados al inicio y se propondrán una serie de líneas de actuación que se podrían desarrollar en un trabajo futuro.

5.1 Conclusiones

Este trabajo ha estado enfocado principalmente en el análisis de la ontología SSN y su integración en una implementación real. Para poder desarrollar una aplicación que hiciera uso de ella ha sido necesario realizar primero un estudio acerca de diferentes tecnologías relacionadas con las redes de sensores y otros conceptos con los que no se tenía experiencia previa.

Se ha debido abordar el estudio del propio concepto de ontología, incluyendo sus aspectos y elementos más relevantes, como paso previo a la realización propiamente dicha de este Proyecto de Fin de Carrera. El estudio de la ontología SSN y sus posibles aplicaciones resultó más complicado de lo que se estimó en un primer momento debido a que no se encontró demasiada información al respecto.

Como paso previo, al diseño e implementación de la aplicación que se ha desarrollado, y que hace uso de sistemas de sensores modelados en formato OWL, fue necesario realizar la descripción semántica de los dispositivos con los que se ha trabajado a modo de ejemplo.

Las características básicas de estos componentes se establecieron después de una primera toma de contacto con las clases y propiedades que proporciona SSN para la descripción de elementos pertenecientes al ámbito de sensores, pero a lo largo del desarrollo de este trabajo se tuvieron que realizar modificaciones y añadir otras particularidades con las que satisfacer las necesidades de la aplicación final implementada.

Otro elemento en cuyo estudio se ha debido profundizar ha sido en la librería Jena, que ha resultado de gran importancia para el desarrollo de este trabajo debido a la cantidad de funciones que proporciona para el manejo de ontologías.

He conseguido obtener un buen conocimiento de SSN lo cual ha permitido realizar la descripción de un sistema de sensores básico de acuerdo a esta ontología. A pesar de haberse utilizado unos dispositivos básicos a modo de ejemplo, dadas sus características, el modelado realizado permitirá abordar elementos y sistemas más complejos.

Uno de los objetivos que se propusieron al inicio del trabajo y que puede darse como conseguido ha sido la generación semiautomática de código a partir de un modelo de sistemas descrito en función de SSN. Haciendo uso de clases y propiedades de la ontología ha sido posible la generación de interfaces correspondientes a los distintos

tipos de componentes definidos para el desarrollo del proyecto. La implementación de esta aplicación ha servido para entender la forma en la que puede manejarse la información contenida en un fichero OWL.

Por último, implementando manualmente las interfaces obtenidas en el apartado anterior, se ha logrado también desarrollar un programa capaz de representar los distintos dispositivos de un sistema de sensores escrito de acuerdo a la ontología protagonista de este proyecto y presentado en un fichero de tipo OWL.

El desarrollo de este Proyecto de Fin de Carrera me ha resultado sumamente interesante pues me ha permitido trabajar en torno a ámbitos de conocimiento que no se conocían anteriormente.

5.2 Trabajos futuros

A continuación se proponen algunas medidas con las que se podría mejorar la aplicación desarrollada y la descripción de los dispositivos.

- Generalizar la aplicación correspondiente a la creación del modelo de sensores para que puedan definirse sensores de distintos tipos.
  De la forma en la que se ha desarrollado esta aplicación, sólo es posible definir el nombre con el que se representará dentro del modelo el sensor que se está creando. De manera automática se generan las propiedades que lo identifican como un sensor de temperatura e incluso se establecen sus rangos de medición, debido a las características de los métodos que se han implementado para la creación de los dispositivos. Cambiando los parámetros de entrada y el funcionamiento interno de estos métodos sería posible personalizar muchos de los atributos que se aplican por defecto a los sensores. Incluso suponiendo que únicamente se quisieran modelar sensores de temperatura, podría añadirse una nueva funcionalidad a la aplicación que posibilitara la creación de sensores con distintos rangos de medición.

Otra consideración a tener en cuenta en relación con esta aplicación sería la posibilidad de cargar previamente un modelo OWL en el que se encontraran definidos los distintos tipos de dispositivos que pudieran crearse y a partir de las características extraídas de este modelo, solicitar al usuario los campos necesarios de acuerdo al componente que se quisiera describir.

- Definir subclases de *SensingDevice* para cada tipo de sensor que se pueda modelar al realizar la generalización. De este modo, el nombre de la clase a la que pertenezcan las instancias una vez creadas, sería suficiente para determinar el tipo de dispositivo que es. Como vimos en el apartado relativo a la descripción del sistema de sensores, al crear todos los *individuals* a partir de la clase *SensingDevice*, no se tenía información suficiente acerca del tipo de sensor que representaba. Es importante comentar que además de crear subclases como *TemperatureSensor*, *WindSensor*, etc. es necesario relacionar los *individuals* con propiedades adicionales que aporten más información relativa a las características del sensor que un simple nombre.

- Desarrollar la aplicación para la representación de los dispositivos en forma de servidor, e implementar un cliente que haga uso de dicho servidor.

- Definición de una ontología de servicios o utilización de una existente con la que integrar la ontología SSN para la implementación de aplicaciones finales.

6 Referencias bibliográficas

[1] W3C Semantic Sensor Network Incubator Group, "The SSN Ontology," 2012.

[2] SSN Ontology. [Online]. http://www.w3.org/2005/Incubator/ssn/ssnx/ssn

[3] Oktie Hassanzadeh , "Introduction to Semantic Web Technologies & Linked Data," University of Toronto, 2011.

[4] Semantic Web. [Online]. http://www.princeton.edu/~achaney/tmve/wiki100k/docs/Semantic_Web.html

[5] Toby Segaran, Colin Evans, and Jamie Taylor, *Programming the semantic web*.: O'Reilly Media, Inc., 2009.

[6] Ivan Herman. Why OWL and not WOL? [Online]. http://www.w3.org/People/Ivan/CorePresentations/RDFTutorial/Slides.html#(114)

[8] Deborah L. McGuinness and Frank Van Harmelen. OWL Web Ontology Language Overview. [Online]. http://www.w3.org/TR/2004/REC-owl-features-20040210/

[7] Joshua Tauberer. O'Reilly XML.com. [Online]. http://www.xml.com/pub/a/2001/01/24/rdf.html?page=2

[9] Semantic Sensor Network Incubator Group. W3C Semantic Sensor Network Incubator Group. [Online]. http://www.w3.org/2005/Incubator/ssn/wiki/Incubator_Report#Reviewed_ontologies

[10] Susel Fernández, Iván Marsa-Maestre, Juan R. Velasco, and Bernardo Alarcos, "Ontology Alignment Architecture for Semantic Sensor Web Integration," Department of Computing Engineering, University of Alcala, 2013.

[11] Danh Le Phuoc and Laurent Lefort. Review of Sensor and Observation ontologies. [Online]. http://www.w3.org/2005/Incubator/ssn/wiki/Incubator_Report#OntoSensor

[12] Rimel Bendadouche, Catherine Roussey, Gil De Sousa, Jean-Pierre Chanet, and Kun Mean Hou, "Extension of the SSN Ontology for Wireless Sensor Networks: The Stimulus-WSNnode-Communication Pattern,".

[13] Viviana Mascardi, Valentina Cordi, and Paolo Rosso, "A Comparison of Upper Ontologies,".

[14] Pratikkumar Desai, Cory Henson, Pramod Anatharam, and Amit Sheth, "SECURE - Semantics Empowered resCUe Environment," Kno.e.sis - Ohio Center of Excellence in Knowledge-enabled Computing,.

[15] Spitfire - Semantic Web interaction with Real Objects. [Online]. http://spitfire-project.eu/

[16] Semantic Sensor Grids for Environmental Applications. [Online]. http://www.semsorgrid4env.eu/

[18] Apache Jena. [Online]. http://jena.apache.org/documentation/ontology/

[17] Swiss Experiment Platform. [Online]. http://www.swiss-experiment.ch/start/index.html

7 Anexo

Fichero OWL generado por la aplicación con la representación de un *SensorDummy*.

Este anexo incluye únicamente algunos fragmentos del fichero OWL correspondiente a un sistema de sensores muy sencillo, formado por un solo *Dummy*.

En concreto se indican algunas de las propiedades, *individuals* y clases que se han creado y utilizado para la representación del *Dummy* y sus componentes.

El espacio de nombres definido para las instancias generadas es: "http://www.ejemplo.com/pfc#"

El orden en el que se han ido presentando los fragmentos no corresponde al orden real en el que aparecen en el documento. Al inicio se colocado la definición del *dummy*.

```
<?xml version="1.0" encoding="MacRoman"?>
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:skos="http://www.w3.org/2004/02/skos/core#"
   xmlns:cc="http://creativecommons.org/ns#"
   xmlns:owl="http://www.w3.org/2002/07/owl#"
   xmlns:dc="http://purl.org/dc/elements/1.1/"
   xmlns:dct="http://purl.org/dc/terms/"
   xmlns:DUL="http://www.loa-cnr.it/ontologies/DUL.owl#"
   xmlns:j.0="http://www.ejemplo.com/pfc#"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xmlns:ssn="http://purl.oclc.org/NET/ssnx/ssn#" >

                                    [ . . .]


  <rdf:Description rdf:about="http://www.ejemplo.com/pfc#SensorDummy1">
   <ssn:implements rdf:resource="http://www.ejemplo.com/pfc#getLocation"/>
   <ssn:implements
rdf:resource="http://www.ejemplo.com/pfc#listComponents"/>
   <ssn:hasSubSystem
rdf:resource="http://www.ejemplo.com/pfc#LightDevice1"/>
   <ssn:hasSubSystem
rdf:resource="http://www.ejemplo.com/pfc#TemperatureSensor1"/>
   <DUL:hasLocation>Diatel EUITT</DUL:hasLocation>
   <rdf:type rdf:resource="http://purl.oclc.org/NET/ssnx/ssn#System"/>
  </rdf:Description>

                                    [ . . .]

<rdf:Description rdf:about="http://www.ejemplo.com/pfc#TemperatureSensor1">
   <ssn:implements rdf:resource="http://www.ejemplo.com/pfc#getProperties"/>
   <ssn:implements rdf:resource="http://www.ejemplo.com/pfc#getTemperature"/>
   <ssn:implements rdf:resource="http://www.ejemplo.com/pfc#getMinRange"/>
   <ssn:implements rdf:resource="http://www.ejemplo.com/pfc#getMaxRange"/>
   <ssn:observes rdf:resource="http://www.ejemplo.com/pfc#Temperature"/>
   <ssn:hasMeasurementCapability
rdf:resource="http://www.ejemplo.com/pfc#TempMeasurementCapability"/>
   <rdf:type rdf:resource="http://purl.oclc.org/NET/ssnx/ssn#SensingDevice"/>
  </rdf:Description>

                                    [ . . .]

  <rdf:Description rdf:about="http://www.ejemplo.com/pfc#LightDevice1">
   <ssn:implements rdf:resource="http://www.ejemplo.com/pfc#setState"/>
```

```
   <ssn:implements rdf:resource="http://www.ejemplo.com/pfc#setLightIntesity"/>
   <ssn:implements rdf:resource="http://www.ejemplo.com/pfc#getState"/>
   <ssn:implements rdf:resource="http://www.ejemplo.com/pfc#getLightIntesity"/>
   <ssn:implements rdf:resource="http://www.ejemplo.com/pfc#DeviceProcess"/>
   <rdf:type rdf:resource="http://purl.oclc.org/NET/ssnx/ssn#Device"/>
  </rdf:Description>

                                        [ . . .]

<rdf:Description rdf:about="http://www.ejemplo.com/pfc#getLocation">
   <ssn:implementedBy
rdf:resource="http://www.ejemplo.com/pfc#SensorDummy1"/>
   <DUL:hasDataValue>public String getLocation(OntModel model, String
dummyName);</DUL:hasDataValue>
   <rdf:type rdf:resource="http://www.loa-cnr.it/ontologies/DUL.owl#Method"/>
  </rdf:Description>
                                        [ . . .]

<rdf:Description rdf:about="http://www.ejemplo.com/pfc#listComponents">
   <ssn:implementedBy
rdf:resource="http://www.ejemplo.com/pfc#SensorDummy1"/>
   <DUL:hasDataValue>public String[] listComponents(OntModel model, String
dummyName);</DUL:hasDataValue>
   <rdf:type rdf:resource="http://www.loa-cnr.it/ontologies/DUL.owl#Method"/>
  </rdf:Description>

                                        [ . . .]

  <rdf:Description rdf:about="http://www.ejemplo.com/pfc#LightIntensity">
   <rdf:type rdf:resource="http://purl.oclc.org/NET/ssnx/ssn#Output"/>
  </rdf:Description>
                                        [ . . .]

  <rdf:Description rdf:about="http://www.ejemplo.com/pfc#State">
   <rdf:type rdf:resource="http://purl.oclc.org/NET/ssnx/ssn#Output"/>
  </rdf:Description>


<rdf:Description rdf:about="http://www.ejemplo.com/pfc#Temperature">
   <rdf:type rdf:resource="http://purl.oclc.org/NET/ssnx/ssn#Property"/>
   <rdfs:subClassOf
rdf:resource="http://purl.oclc.org/NET/ssnx/ssn#ObservationValue"/>
   <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
  </rdf:Description>

                                        [ . . .]
```

```
 <rdf:Description rdf:about="http://www.ejemplo.com/pfc#TemperatureValue">
  <DUL:isClassifiedBy rdf:resource="http://www.ejemplo.com/pfc#Celsius"/>
  <rdf:type rdf:resource="http://www.ejemplo.com/pfc#Temperature"/>
 </rdf:Description>

                                    [ . . .]

 <rdf:Description rdf:about="http://www.ejemplo.com/pfc#RangeMaxValue">
  <DUL:hasDataValue
rdf:datatype="http://www.w3.org/2001/XMLSchema#float">100.0</DUL:hasDataV
alue>
  <DUL:isClassifiedBy rdf:resource="http://www.ejemplo.com/pfc#Celsius"/>
  <rdf:type rdf:resource="http://www.ejemplo.com/pfc#Temperature"/>
 </rdf:Description>

                                    [ . . .]

 </rdf:Description>
 <rdf:Description rdf:about="http://www.ejemplo.com/pfc#RangeMinValue">
  <DUL:hasDataValue
rdf:datatype="http://www.w3.org/2001/XMLSchema#float">-
15.0</DUL:hasDataValue>
  <DUL:isClassifiedBy rdf:resource="http://www.ejemplo.com/pfc#Celsius"/>
  <rdf:type rdf:resource="http://www.ejemplo.com/pfc#Temperature"/>
 </rdf:Description>
                                    [ . . .]

<rdf:Description
rdf:about="http://www.ejemplo.com/pfc#TempMeasurementCapability">
  <ssn:hasMeasurementProperty
rdf:resource="http://www.ejemplo.com/pfc#TempMeasurementRange"/>
  <rdf:type
rdf:resource="http://purl.oclc.org/NET/ssnx/ssn#MeasurementCapability"/>
 </rdf:Description>

                                    [ . . .]

 <rdf:Description
rdf:about="http://www.ejemplo.com/pfc#TempMeasurementRange">
  <j.0:hasMeasurementPropertyMinValue
rdf:resource="http://www.ejemplo.com/pfc#RangeMinValue"/>
  <j.0:hasMeasurementPropertyMaxValue
rdf:resource="http://www.ejemplo.com/pfc#RangeMaxValue"/>
  <rdf:type
rdf:resource="http://purl.oclc.org/NET/ssnx/ssn#MeasurementRange"/>
 </rdf:Description>
```

[. . .]

```
 <rdf:Description rdf:about="http://www.ejemplo.com/pfc#DeviceProcess">
  <ssn:hasOutput rdf:resource="http://www.ejemplo.com/pfc#LightIntensity"/>
  <ssn:hasOutput rdf:resource="http://www.ejemplo.com/pfc#State"/>
  <rdf:type rdf:resource="http://purl.oclc.org/NET/ssnx/ssn#Process"/>
 </rdf:Description>
```

[. . .]

```
<rdf:Description rdf:about="http://www.ejemplo.com/pfc#TemperatureOutput">
  <ssn:hasValue rdf:resource="http://www.ejemplo.com/pfc#TemperatureValue"/>
  <rdf:type rdf:resource="http://purl.oclc.org/NET/ssnx/ssn#SensorOutput"/>
 </rdf:Description>
```

[. . .]

```
rdf:Description
rdf:about="http://www.ejemplo.com/pfc#hasMeasurementPropertyMaxValue">
  <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-
ns#Property"/>
 </rdf:Description>
```

[. . .]

```
<rdf:Description rdf:about="http://www.ejemplo.com/pfc#getMaxRange">
  <ssn:implementedBy
rdf:resource="http://www.ejemplo.com/pfc#TemperatureSensor1"/>
  <DUL:hasDataValue>public float getMaxRange(OntModel
model);</DUL:hasDataValue>
  <rdf:type rdf:resource="http://www.loa-cnr.it/ontologies/DUL.owl#Method"/>
 </rdf:Description>
```

[. . .]

```
<rdf:Description rdf:about="http://www.ejemplo.com/pfc#getMinRange">
  <ssn:implementedBy
rdf:resource="http://www.ejemplo.com/pfc#TemperatureSensor1"/>
  <DUL:hasDataValue>public float getMinRange(OntModel
model);</DUL:hasDataValue>
  <rdf:type rdf:resource="http://www.loa-cnr.it/ontologies/DUL.owl#Method"/>
 </rdf:Description>
```

[. . .]

```
 <rdf:Description rdf:about="http://www.ejemplo.com/pfc#getLightIntesity">
```

```
   <ssn:implementedBy
rdf:resource="http://www.ejemplo.com/pfc#LightDevice1"/>
   <DUL:hasDataValue>public float getLightIntesity();</DUL:hasDataValue>
   <rdf:type rdf:resource="http://www.loa-cnr.it/ontologies/DUL.owl#Method"/>
  </rdf:Description>


  <rdf:Description rdf:about="http://www.ejemplo.com/pfc#setLightIntesity">
   <ssn:implementedBy
rdf:resource="http://www.ejemplo.com/pfc#LightDevice1"/>
   <DUL:hasDataValue>public void setLightIntesity(float
intensity);</DUL:hasDataValue>
   <rdf:type rdf:resource="http://www.loa-cnr.it/ontologies/DUL.owl#Method"/>
  </rdf:Description>


  <rdf:Description rdf:about="http://www.ejemplo.com/pfc#setState">
   <ssn:implementedBy
rdf:resource="http://www.ejemplo.com/pfc#LightDevice1"/>
   <DUL:hasDataValue>public void setState(boolean state);</DUL:hasDataValue>
   <rdf:type rdf:resource="http://www.loa-cnr.it/ontologies/DUL.owl#Method"/>
  </rdf:Description>
```